# Spring Project Developer Documentation

## Index



**This document describes “guide lines” with the intention to give the project transparent processes. These processes gives some structure that can support the development process. But situations differ and things change with time, so only apply this as a “rule” when it makes sense to do so.**

## Spring Developers

Spring is a open source project where programmers and artists work together to develop a great gaming experience. To effectively manage the project the people involved with Spring are described in several broad groups. The first rough distinction is people that develop things and those that play it. This document is about those that develop.

Of those people that develop we have another distinction in developers that focus on code and developers that focus on content and within these two groups there are again sub-groups. Depending on your interests you will fit in one or more of these sub-groups and with that your requirements for the development environment.

At this moment this document describes several types of developers “AI developers” and “MOD developers” are more or less done, the description for “SVN Administrators”, “Engine Developers” and “Release Manager” still needs work but the chapters are there.

For Spring developers most things are the same whether you write code or make content so most information can be fond in this general chapter with some specific things described in a chapter for the type of development where your interested in.

### How do I become a Spring Developer?

You have to convince us that you know your stuff and can contribute to the project.

You have to release your work under a open source license (for the spring engine the GPL see http://www.gnu.org/copyleft/gpl.html) and please read the license as it has some requirements that you should understand.
If your work is not “linking” the engine you can use several other licenses see http://www.opensource.org/licenses/ as required by the licenses you upload your source files. What the source of the files is depends on what you created.

Next to that you must like to continue the work you're doing within the project. The development environment is intended for people that make continued use of it.

You must also understand that the right to commit to svn is only intended to help make the development process easier. Having your work included in svn won't guarantee that it will also be included with a Spring release. Stability concerns, file size restrictions, system requirements, lack of an active maintainer and **many** more things can make the people that do the release decide to exclude it.
Just like the right to post on the forum won't necessarily make people agree with your idea's.

**Last but not least, you must be a nice person to hang out with, people that are bitching or irritating won't be given any rights.**

### How do I retire as a Spring Developer?

We really do love you!
And we like to have many people join in and help make Spring the best it can be. But sometimes people just don't “fit in” any more, write awful code, have gone missing, have found a new challenge.. and having a little procedure to help clean things up a bit can really be refreshing to the project.

First off, at a minimum you need to contribute something (a fix, feature or update etc.) every three months. If you're inactive for three months you can expect to be contacted by another developer with the “You have been inactive for some time in the Spring project. How are things going over there?” question.
If you don't reply to these mails you can be removed as an active developer after a few weeks. If you do reply and intent to become an active developer again you can keep your status as an active developer for another three months. If you did not commit anything useful after this time you can lose you commit rights, that is until you have some useful contributions.

Now there is also the possibility that you do contribute useful things to the project but are a pain to work with. As we consider “having a good time” more important then just about anything, you would have a real problem. In other words, you could be the pinnacle of Spring development but if we don't like you, you're still out.

## Subversion

Spring uses Subversion (svn) to manage the projects (source) files. You can see it as a large shared directory where all developers put there files in. As a developer you gain the right to change the files in this shared directory.
Putting your changes in svn is called “committing” to svn. Having “commit” rights brings with it a responsibility.

There are some general rules that help you manage this responsibility.

- Don't commit any OTA, SupCom or other stuff that is not distributed under a open license.
- Don't commit files like “Thumbs.db”.
- Avoid committing generated data to svn (logfiles, unit or map lists, etc.)
- Don't duplicate code or data.
- Don't break the build. (that means it should always be in a working state)
- Don't commit large files.
- Add descriptive commit messages.

Look here for some examples about good svn use: http://www.t2-project.org/svn-commit-tutorial.html at some point we could write or own documentation on this.

## Directory structure

The stuff in svn is spread out over several directories, the main directories are:

### trunk/

This directory is mainly intended for current development, if people ask you to compile the engine from “head” they mean you should compile the source found in the “rts” subdirectory. The other directories inside trunk are intended for other things that need to be developed.
Trunk is intended constant development of code.

### mods/

Mod development can take place here, each mod has its own subdirectory with a trunk, branches and other project directories.
The mods/*mod name*/trunk directory is intended for constant development of the mod.

### branches/

Sometimes there is a need to split off a copy of trunk to take development in a specific direction. An important branch is the “release branch” which is explained further down. Normally branches are created when a developer needs to alter a lot of code that might break things like with porting the engine to a different OS or to implement a new subsystem like a new animation system.

### tags/

Here you find the source for a specific Spring release, it's important for GPL compliance but its main use is to see if certain code was included with a specific release.

## Release branch

The release branch is split from trunk when we like to do a new release. That way the code can be tested and stabilized, once everything is ready the Release Manager releases a new version and we sit back while our server gets itself ddos'ed.

The release branch is “feature-frozen”, that means only bug fixes can be committed.

This way we hope to:
* provide a release without bugs introduced by last-minute features / code redesigns / etc.,
* have more time for testing & bugfixing, without having to bother about a constant inflow of new bugs.
* give external (read: without commit access) content- and AI developers some time to prepare their stuff for the release.

You probably won't get the right to commit everywhere in svn, this has some advantages in that your limited in what you can do wrong when you're new to svn. But the main advantage is that it ensures people first talk about their intentions and that they understand the specifics of the development role they embark on.
So if you're an engine developer and you would like to do some MOD development to you will have to tell the other developers your intentions and show that you know some things about MOD development.
This process is here to prevent situations where MOD developers commit ugly code to the Spring engine and the other way around prevent engine developers from changing the balance of a mod.
Commit restrictions should never become a problem, if you need (or just like) more commit rights ask around and if you promise to use the rights with care a lot is possible. But be aware that you are responsible for any commits you make to svn.

The SVN Administrators have the right to reorganise the svn, give or retract commit access to different parts of svn and just generally change things to make development easier.

If you like to see something in svn reorganised talk with the SVN Administrators, we all like to make things work as smooth as possible and your feedback can help shape things.

Then we still have the question on how to actually use svn as a developer.
The long answer: http://svnbook.red-bean.com/en/1.1/ch01.html
The more specific answer http://svnbook.red-bean.com/en/1.1/ch03s05.html
(We need some more documentation on how to use Springs svn on the different platforms.)

## Forum

Spring has a forum at http://taspring.clan-sy.com/phpbb/index.php all of the sub-forums are open for people to post in except for the mapping sub-forum.
This document presumes that we make another sub-forum where only Spring developers can post. This to keep communication lines short and prevent the gamers and other non-contributors from derailing important discussions.
Also a smaller group of individuals is easier to mediate with disagreements.

## Engine Development

This chapter is mostly a place holder.

Release branch:
* Always separate bugfix commits and commits that add new features/code!
* Mark your bugfix commits with the text "BUGFIX" (in caps, without quotes) somewhere in your commit message. there is a little script to (semi-)automagically perform the merges, so there's a good chance forgetting this will not get your bugfixes in the release, unless you merge them manually.

## AI Development

Development of AI's is very interconnected with mod and engine development. Having easy access to these dependency's can greatly improve the AI development process. This chapter describes some specifics things with AI development.

Being an AI developer will mean you can commit to the directories concerned with AI development. If you like to commit to other parts of the project you will have to be accepted for that type of development.

At this moment the SVN Administrators decided on the following rules regarding AI development:

- Don't update or modify other AI's in svn without first asking the author(s). This applies to GroupAI's too.
- Don't bloat the svn with support for older versions of Spring or its mods.
- Try to make sure your AI works on all supported platforms. (use the buildbot)
- Do not import or start a new AI (Group or Global) without permission of the SVN Administrators.

Release branch:
The AI in the release branch, will be the AI distributed with the release.
You can use the release branch to stabilize your code for release but make sure (at

a minimum) to merge a stable revision from trunk into the release branch any time between the creation of that branch and the actual release.

## MOD Development

Development of mod's can really benefit from teamwork and close integration with engine and AI development. As mod's can be relativity small and as they consist for a large part out of text files Spring's svn can be a real enabler to lift mod development to a new level. This chapter describes some things about MOD development within the Spring Project.

Being a MOD developer will mean you can commit to the directories concerned with MOD development. If you like to commit to other parts of the project you will have to be accepted for that type of development.

The “rules” so far are:

- Don't use **ANY** content that is not licensed under an open source license.
- Your mod can't use trademarks or be (heavily) based on the "intellectual property" of another party.
- Don't use units from another mod with different statistics. ??
- Don't upload large source files to svn but put them on WebDav. (This needs to be set up first)
- Do not import or start a new mod without permission of the SVN Administrators.

If you don't fit in these criteria but are a capable and active developer of a Spring mod (for example your mod uses trademarks of another party) then your mod can't be developed in the Spring svn, but your are still invited to join in on the discussions on the sub-forum for Spring developers. (See chapter “Forum”)
Having your mod files available under an open source license for use in other mods would still be nice. Your mod can also use content from the Spring svn then.

## SVN Administrators

This is mostly a place holder.

SVN Administrators manage the development environment.
Two things included in this are managing the commit rights and the directory structure in SVN.
Officially Jelmer, Tobi and Chris are SVN Administrators but Chirs is inactive at this moment.

## Release Manager

This is mostly a place holder.

Release manager maintains the release branches and releases new Spring versions.
Jelmer is Springs Release Manager.

| *Version* | *Date* | *Author* | *Note* |
|---|---|---|---|
| 0.1 | 2006-07-05 | Tim Blokdijk | Initial version |
| 0.2 | 2006-07-06 | Tim Blokdijk | Feedback from Tobi Vollebregt |
| 0.3 | 2006-07-10 | Tim Blokdijk | Feedback from Tom Nowell |
| 0.4 | 2006-08-04 | Tim Blokdijk | Rewrote the document to make it apply to Spring Development in general. |
| 0.5 | 2006-08-04 | Tim Blokdijk | Added chapter “Forum and MOD Development” removed “Random questions”. |
| 0.6 | 2006-08-05 | Tim Blokdijk | Fixed a few grammar mistakes. |
| 0.7 | 2006-08-05 | Tim Blokdijk | A little feedback from “Declan” about binary file size restrictions. |
| 0.8 | 2006-08-05 | Tim Blokdijk | Grammar corrections from Tobi Vollebregt. |
| 0.9 | 2006-08-27 | Tim Blokdijk | Included an explanation of the release branches. |
| 0.10 | 2006-08-28 | Tim Blokdijk | Split the subversion chapter in several parts and made a description of dirs. |
| 0.11 | 2006-08-28 | Tim Blokdijk | Started “Engine Development”, “Release Manager” and “SVN Administrators”. |