Fossil Version Control
A Users Guide

Jim Schimpf

Document Number: PAN-20100424
Revision Number: 0.2
24 April 2010

Pandora Products.
215 Uschak Road
Derry, PA 15627

Pandora Products.

215 Uschak Road

Derry, PA 15627

Phone: 724.539.1276

Fax: 724.539.1276

Web: http://pandora.dyn-o-saur.com:8080/cgi-bin/Book.cgi

Email: jim.schimpf@gmail.com

Document Revision History

Use the following table to track a history of this documents revisions. An entry should be made into this table for each version of the document.

Version	Author	Description	Date
0.1	js	Initial Version	24-Apr-2010
0.2	js	Finishing up Single User Chapter	27-Apr-2010
0.3	js	Working on introduction chapter	30-Apr-2010

Contents

List of Figures

Foreward

1 Source Control & Why you need it

1.1 What is it

A source control system is software that manages the files in a project. A project (like this book or software application) usually has a number of files. You then put all these files in directory and at times subdivide even that with subdirectories. At any particular time this set of files in their present edited state make up a working version of the project. If other people are using the project you give them this and as usually is the case they find problems and you fix them. This results is another version of the project slightly changed which goes through the cycle again. (This is why books have editions and software has versions...)

Software developers on large projects with multiple developers could see this cycle and realized they needed a tool to control the changes. With multiple developers sometimes the same file would be edited by two different people changing it in different ways or records of what got changed got lost and it was hard to bring out a new release of the software so that all the bugs were fixed and enhancements made.

A tool called Source Code Control System [5] was developed at Bell Labs in 1972 to track changes in files. It would remember each of the changes made to a file and store comments why this was done. It also limited who could edit the file so conflicting edits would not be done. [4]

This was important but developers could see more was needed, they needed to be able to save the state of all the files in a project and give it a name (i.e. Release 3.14). As software projects mature you will have a released version of the software being used and having bug reports written against it and the next release of the software is probably being developed adding features. The source control system would have to handle what are now called branches where you have say Version 1 which is released but having fixes added to create Version 1.1, 1.2, etc. While you also have Version 2 with new features added under work.

In 1986 the open source Concurrent Version Control system CVS [2] was developed. This system could label groups of files and allow multiple branches (i.e. versions) simultaneously. There have been many other systems developed since them some open source and some proprietary.

Fossil was originally released in 2006 [3] is an easy to install version control system that also includes a trouble ticketing system (Figure 17 on page 15), a wiki (Figure 2.6 on page 18) and self hosted web server (Figure 5 on page 6).

1.2 Why do it ?

Why do you want to use a source control system ? To use one restricts your freedom, you won't be able to create or delete files at random, move files between directories at random. Making changes in your code becomes a check list of steps and must be followed carefully.

With all those hassles why do it. The biggest answer if freedom (huh ?). By following the proceedure of a source control system you gain the freedom to modify your code any way you want. How does that follow ? One of the most horrible feelings as a developer is the "It worked yesterday" syndrome. That is you had code that worked just fine and now it doesn't. You have a very helpless feeling of how do you get back to working code. With a source control system and careful adherence to proceedures you can just go back in time and get yesterdays code and then starting from known good code you can figure out what happened.

Having a source control system also gives you the freedom to experiment, let's try that radical new technique and if it doesn't work it's easy to just go back to what we had.

The rest of this book is a user manal for the Fossil version control system that does the code management and much much more. It is also simple, runs on multiple OS's and is FREE. It is simple to install as it has only one executable and the repositories it creates are a single file that is easy to back up and is usually only 50% the size of the original source.

1.3 Source control description

1.3.1 File Check Out

First source control system in large use

1.3.2 Merge

Early version of this would be

2 Single User Fossil Use

2.1 Introduction

If you have read this far and are at least persuaded to try, you will want to put one of your projects under software control using Fossil. This chapter is set up to lead you through that task and show you how to adapt your development to using this tool. An assumption is made that you will be the only person using the repository, you are the designer, developer and maintainer of this project. After you are comfortable using the tool the next chapter will show how you use it when you have multiple people working on a project.

2.2 Creating a repository

2.2.1 Introduction

In the spirit of "eating one's own dogfood" we will use this book as the project we are going to manage with Fossil. The book is a directory of text files (we are writing it using LyX [1]) and my working area looks like this:

```
FOSSIL/

     This directory holds all my Fossil repositories

FossilBook/

     outline.txt    - Book design
     fossilbook.lyx - The book
     Layout
          fossil.png  - The Fossil logo (image on title page)
     Research
          fossilbib.bib - Working bibliography
          History
               CVC-grune.pdf - Historical paper about CVS
               RCS-A System for Version Control.webloc - RCS bookmark
               SCCS-Slideshow.pdf - Historical paper about SCCS
               VCSHistory -pysync ... .webloc - History of version control
```

This took just a few hours to start preliminary research and build the framework. Since that's about all I'm going to do today I want to build a repository and put all this stuff under Fossil control.

2.2.2 Create Repository

I have a directory called FOSSIL that I keep all my repositories, Fossil doesn't care but it helps me to keep them all in one place so I can back them up. First I need to create a new repository for the book. This is done using the command line after I move into the Fossil book directory.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil new ../FOSSIL/FossilBook.fossil
project-id: 2b0d35831c1a5b315d74c4fd8d532b100b822ad7
server-id:  0149388e5a3109a867332dd8439ac7b454f3f9dd
admin-user: jim (initial password is "ec3773")
```

Figure 1: Create Repository

I create my repositories with the extension .fossil, Fossil again doesn't care but it helps me organize my files. When the create happened it assigned an initial password with an admin user of "jim" (i.e. me). **Save this for later**.

2.2.3 Connect Repository

The repository is created but is empty and has no connection to the book directory. The next step is to open the repository to the book directory with the **open** command.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil open ../FOSSIL/FossilBook.fossil
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil status
repository:   /Users/jschimpf/Public/FOSSIL/FossilBook.fossil
local-root:   /Users/jschimpf/Public/FossilBook/
server-code:  0149388e5a3109a867332dd8439ac7b454f3f9dd
checkout:     279dfecd3f0322f236a92a9a8f3c96acf327d8c1 2010-04-25 12:40:39 UTC
tags:         trunk
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil extra
Layout/fossil.png
Research/History/CVS-grune.pdf
Research/History/RCS--A System for Version Control.webloc
Research/History/SCCS-Slideshow.pdf
Research/History/VCSHistory - pysync - ....webloc
Research/fossilbib.bib
fossilbook.lyx
outline.txt
```

Figure 2: Open Repository & Check

The **open** command seemingly did nothing but checking with the **status** command shows the repository, the directory it's linked to and that we are hooked to the trunk of the store.

The **extra** command shows all the files in the directory that are NOT under control of Fossil. In this case that's all of them since we have not checked in anything.

2.2.4 Add and Initial Commit

I must now add all the relevent files into the repository with the **add** command. The Fossil add is recursive so if I add the top level files it will automaticaly recurse into the subdirectories like Layout and Research and get those files too. Before you do an add it pays to tidy up your directory so you don't accidentally add a bunch of transient files (like object files from a compile). It's easy to remove them but a little tidying before hand can save you some work.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil add .
ADDED  Layout/fossil.png
ADDED  Research/History/CVS-grune.pdf
ADDED  Research/History/RCS--A System for Version Control.webloc
ADDED  Research/History/SCCS-Slideshow.pdf
ADDED  Research/History/VCSHistory - pysync ....webloc
ADDED  Research/fossilbib.bib
fossil: cannot add _FOSSIL_
ADDED  fossilbook.lyx
ADDED  outline.txt
```

Figure 3: Initial file add

I simply told fossil to do an add of the current directory (.) so it got all those files and all the files in the subdirectory. Note the _FOSSIL_ that it didn't add. This is the tag file that fossil keeps in a directory so it knows what repository it belongs to. Fossil won't add this file since it manages it but everything else is fair game.

One final thing before I can quit for the day, these files have been added or rather they will be added to the repository when I commit it. That must be done and we can relax and let Fossil manage things.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil commit -m "Initial Commit"
New_Version: 8fa070818679e1744374bc5302a621490276d739
```

Figure 4: Initial Commit

I added a comment on the commit and it's a good idea to always do this. When later we see the timeline of the commits you will have notes to tell you what was done.

2.2.5 Fossil start up summary

- **fossil create <name>** Creates a new fossil repository
- **fossil open <repository>** While in a source directory connects this directory to the fossil repository
- **fossil add .** Will add (recursively) all the files in the current directory and all subdirectories to the repository
- **fossil commit -m "Initial Commit"** Will put all the currently added files into the repository.

2.3 Set Up User interface

One of the suprising features of Fossil is the webserver. This allows it to have a GUI type user interface with no operating system specific code, the GUI is the web browser supplied by your OS. In the previous steps I checked my project in to a Fossil repository, next I have prep the web interface for production use.

NOTE The Fossil web server uses port 8080 instead of the standard port 80 for all HTTP access. When run it will automatically start your Web browser and open the repository home page. Unfortunately my book work is done on a machine that already has Apache running on 8080 so I will be using port 8081. I will have to add an extra parameter on the UI command line to do this. In most cases this won't be necessary.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil ui -port 8081
```

Figure 5: Starting Webserver

This shows how it's started, as you can see I have picked port 8081 since I am locked out of port 8080. When I do this my browser starts and I am presented with the following home page.

Figure 6: Initial Webserver page

Following the advice on the page I go to setup/config. I am going to do the minimum setup that you should do on all projects. As you get familiar with Fossil you will probably have many more things that you will customize for your taste but what follows are the only things you HAVE to do.

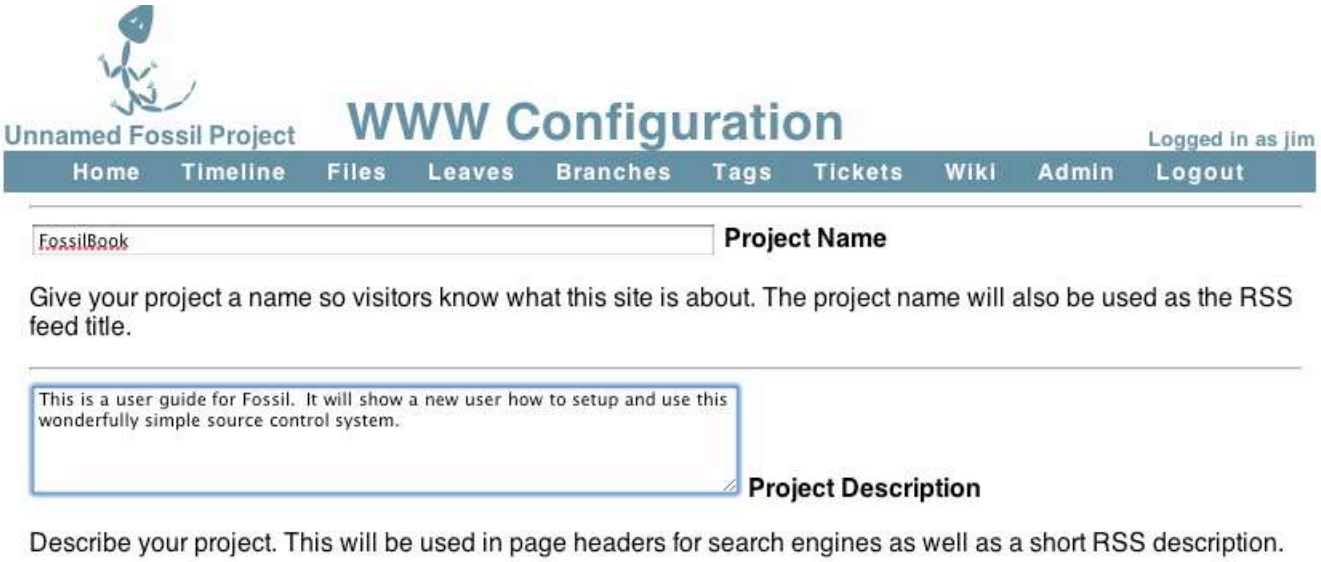

Figure 7: Initial Configuration

I have entered in a project name and put in a description, the project name will be the name of the initial Wiki page (see 2.6 on page 18) and the description is useful for others to see what you are doing here. Then I go to the bottom of the page and pick the **Apply Changes** button.

Next I pick the Admin tab (you can see it in the header bar) and the pick Users from that page. I am presented with the users and will use this to set the password of the project.

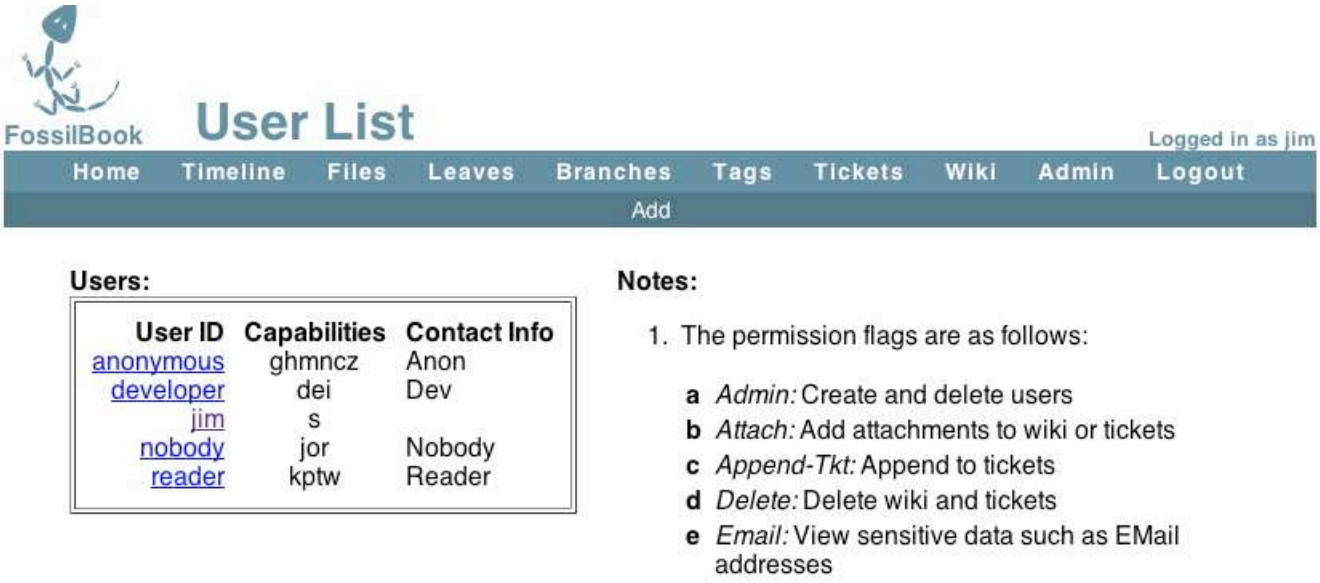

Figure 8: User Configuration

As you can see Fossil automatically configures a number of users beyond just the creator. The anonymous user you have already seen if you went to the Fossil web site to download the code. This user can view and get code but cannot commit code. On the right side of the page are the many options you can give to a user, it's worth reading it all when you set up your repository. The important one is me (jim) which has s or Super User Capabilities. This means I can do anything with the repository.

I will now edit the user Jim to make sure it has the settings I want. In this case you MUST set the password. Remember way back where Fossil set it during the create(2.2.2 on page 4), it's a very good idea to change this to something you can remember rather than the original random one.

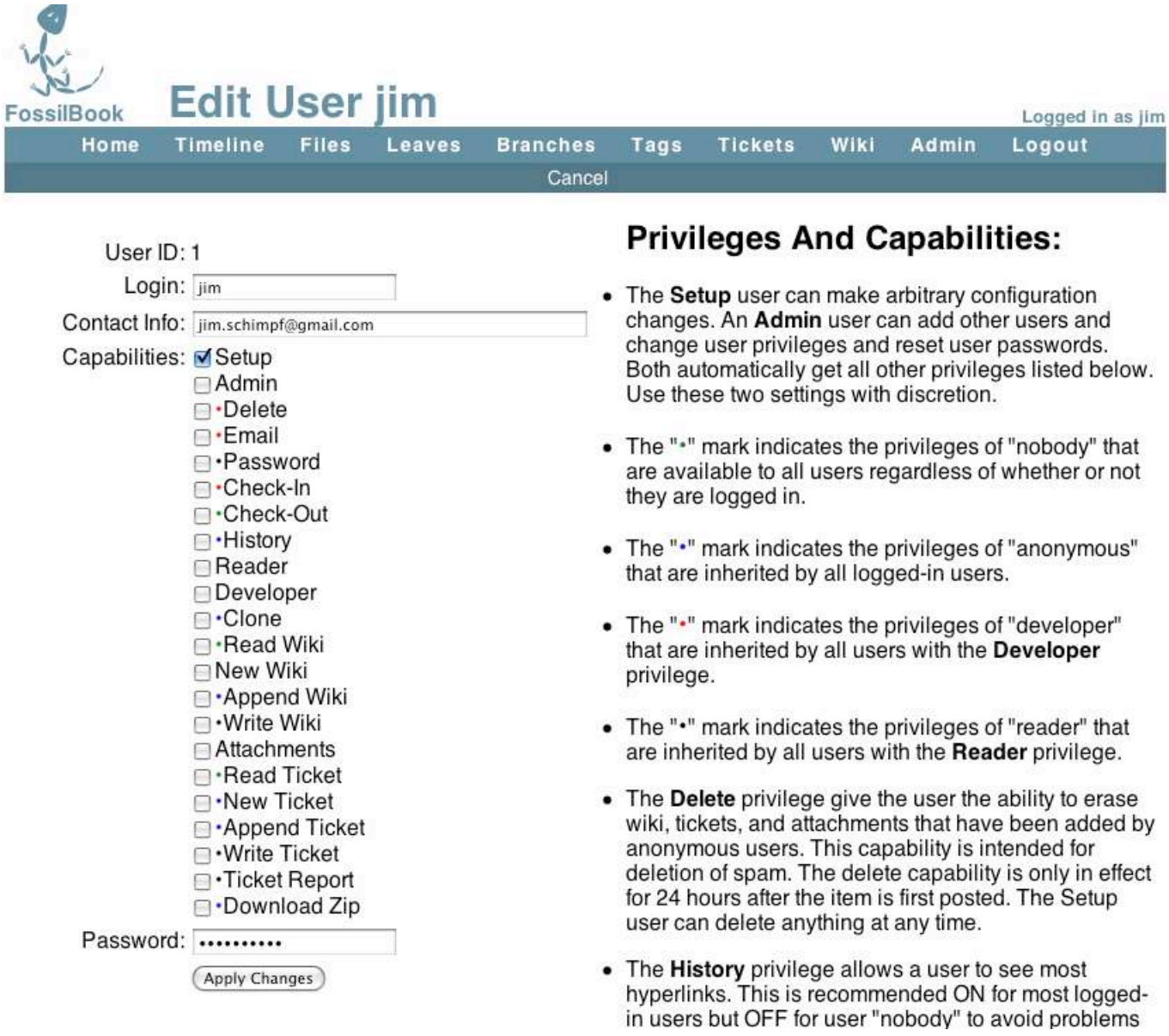

Figure 9: Super User Setup

I have put in my contact information (e-mail address) and while you cannot see it I have typed in a password that I will remember. Then I applied the changes.

Now the repository is ready for further work, it's rather bare bones at this point but the most important things are set up.

2.3.1 User interface summary

- **fossil ui** run in the source directory will start a browser based user interface to fossil.

- **fossil ui -port <IP port #>** Can be used if port 8080 if already in use on your system.
- On the first run it is important to configure your project with a name and set the password for yourself.

2.4 Update Repository

After writing the above section of the book I now have created a bunch of new files and changed some of the existing files in the repository. before quitting for the day I should add these new files into the repository and commit the changes saving this milestone in the project.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil extra
#fossilbook.lyx#
Images/Single_user/config_initial.epsf
Images/Single_user/initial_page.epsf
Images/Single_user/jim_setup.epsf
Images/Single_user/user_config.epsf
fossilbook.lyx~
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil status
repository:   /Users/jschimpf/Public/FOSSIL/FossilBook.fossil
local-root:   /Users/jschimpf/Public/FossilBook/
server-code:  0149388e5a3109a867332dd8439ac7b454f3f9dd
checkout:     8fa070818679e1744374bc5302a621490276d739 2010-04-25 13:09:02 UTC
parent:       279dfecd3f0322f236a92a9a8f3c96acf327d8c1 2010-04-25 12:40:39 UTC
tags:         trunk
EDITED     fossilbook.lyx
```

Figure 10: Project Status

I run fossil extra to see these new files. The image files (those in Images/Single_user) I want to add, the other two files #fossilbook.lyx# and fossilbook.lyx~ I don't want to add since they are temporary artifacts of LyX. I also ran fossil status, this shows changes to files that are already in the repository, there the only files changed are the book text itself, fossilbook.lyx.

All I have to do now is add in the directory Images which will add in the image files I want in the repository. Then I commit the changes to the repository and we can move on to other tasks of the day.

```
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil add Images
ADDED  Images/Single_user/config_initial.epsf
ADDED  Images/Single_user/initial_page.epsf
ADDED  Images/Single_user/jim_setup.epsf
ADDED  Images/Single_user/user_config.epsf
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil commit -m "Initial setup with pictures"
New_Version: a2d12bf532a089ee53578e3e17c6e732c0442f49
```

Figure 11: Update for new files

After doing this commit I can bring up the Fossil ui (see Figure 5 on page 6) and view the project Timeline by picking that tab on the Fossil header. We get this:

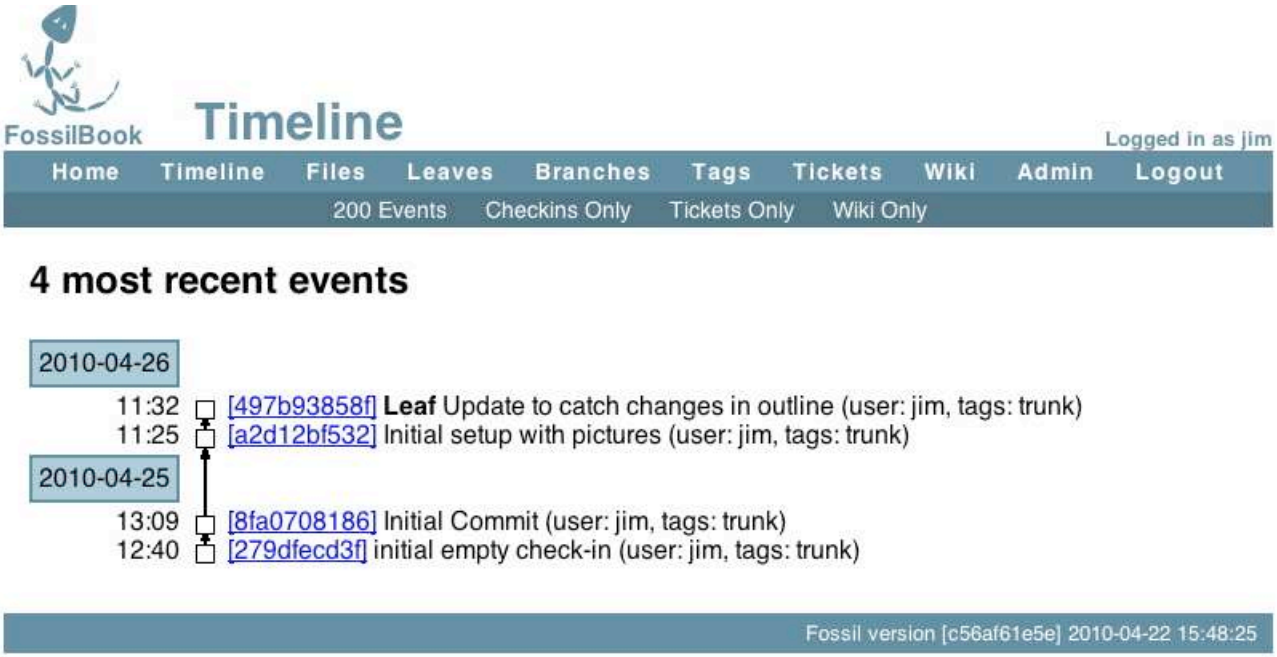

Figure 12: Timeline

You can see all our check-ins thus far and you can see after the check-in from Figure 11 on the preceding page I did another check-in because I missed some changes in the outline. The check-ins are labeled with the first 10 digits of their hash value and these are active links which you can click to view in detail what was changed in that version.

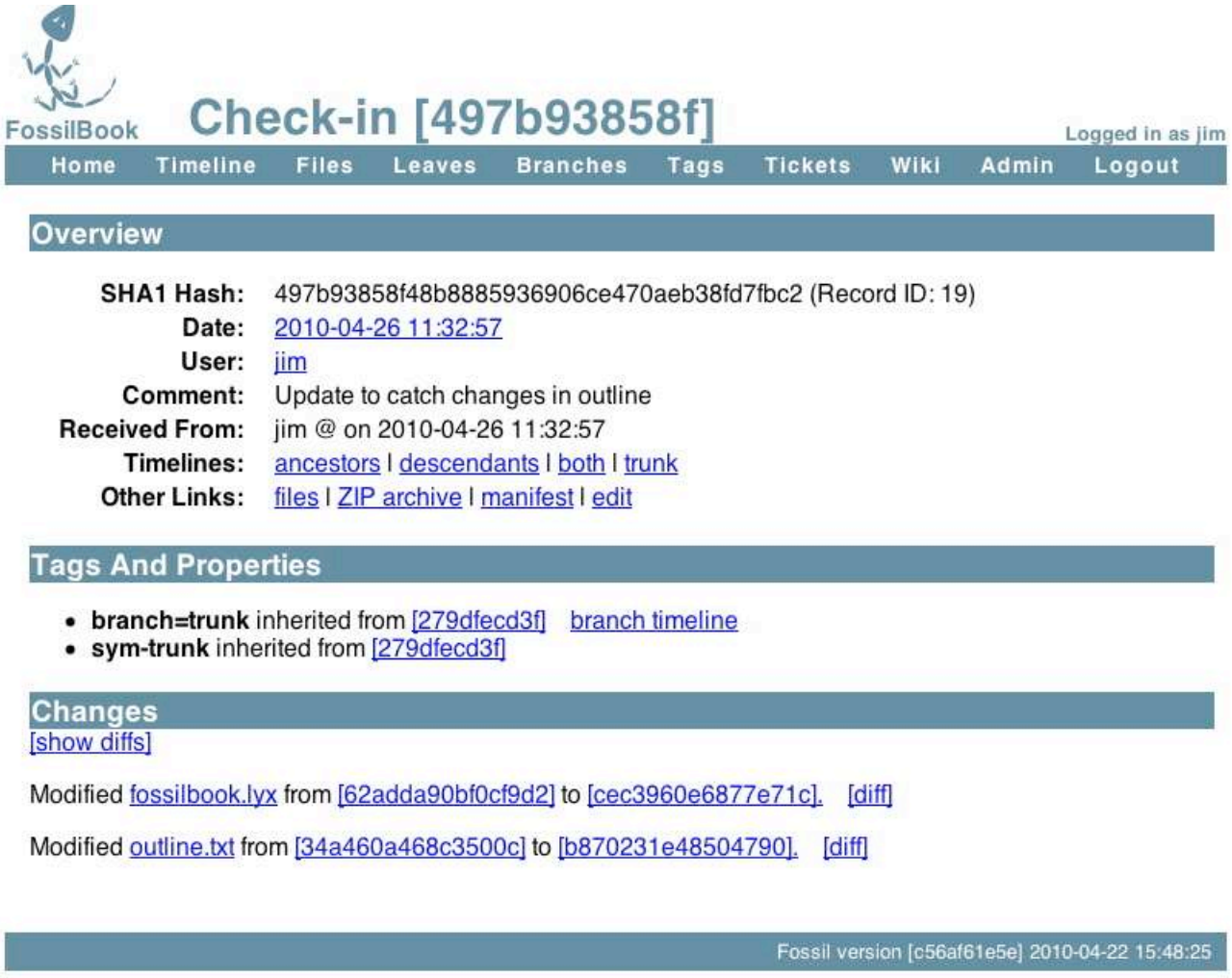

Figure 13: Timeline Detail

I clicked on the very last check-in (the **LEAF**) and the display is shown above. There are many things you can do at this point. From the list of changed files you can pick the diff link and it will show in text form the changes made in that particular file. The Other Links section has a very useful ZIP Archive. Clicking this will download a ZIP of this version to your browser. You will find this useful if you want to get a particular version, in fact this is normally how you get a new version of Fossil from `http://www.fossil-scm.org/`.

2.4.1 Update Summary

- **fossil status** and **fossil extra** will tell you the updated files and files not in the repository before you commit.

- **fossil commit - m "Commit comment"** Commits a changes, it is very important to have a descriptive comment on your commit.

2.5 Tickets

Besides managing your code Fossil has a trouble ticket system. This means you can create a ticket for a problem or feature you are going to add to your system then track your progress. Also you can tie the tickets to specific check-ins of your files. For software this is very useful for bug fixes and feature additions. For example you can look for a bug in the ticket list then have it take you to the change that fixed the problem. Then you know exactly what you did and not have to be confused by other changes you might have made.

When you pick Tickets it will bring up this window. You can create a new ticket, look at the list or generate a new report. Keeping things simple I will just use the All Tickets list for now.

Figure 14: Initial Ticket Window

Picking New Ticket I get a form that I fill out like so:

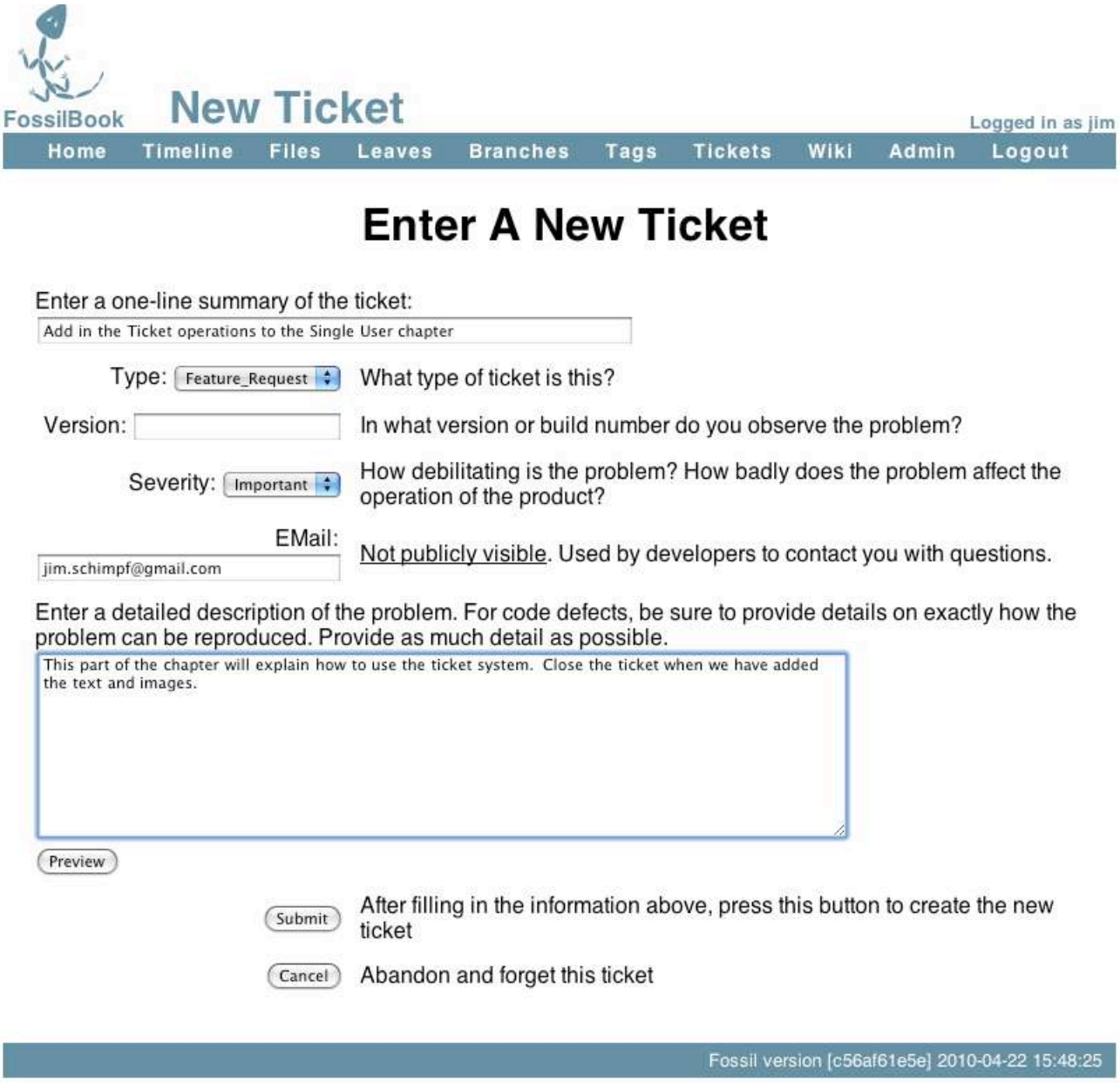

Figure 15: Ticket Form

Pretty simple actually, you can put as much or as little detail in here as you wish but remember this stuff might be really vital 6 weeks or 6 months from now so think of what you would want to know then. When I press Submit I get this showing what I entered.

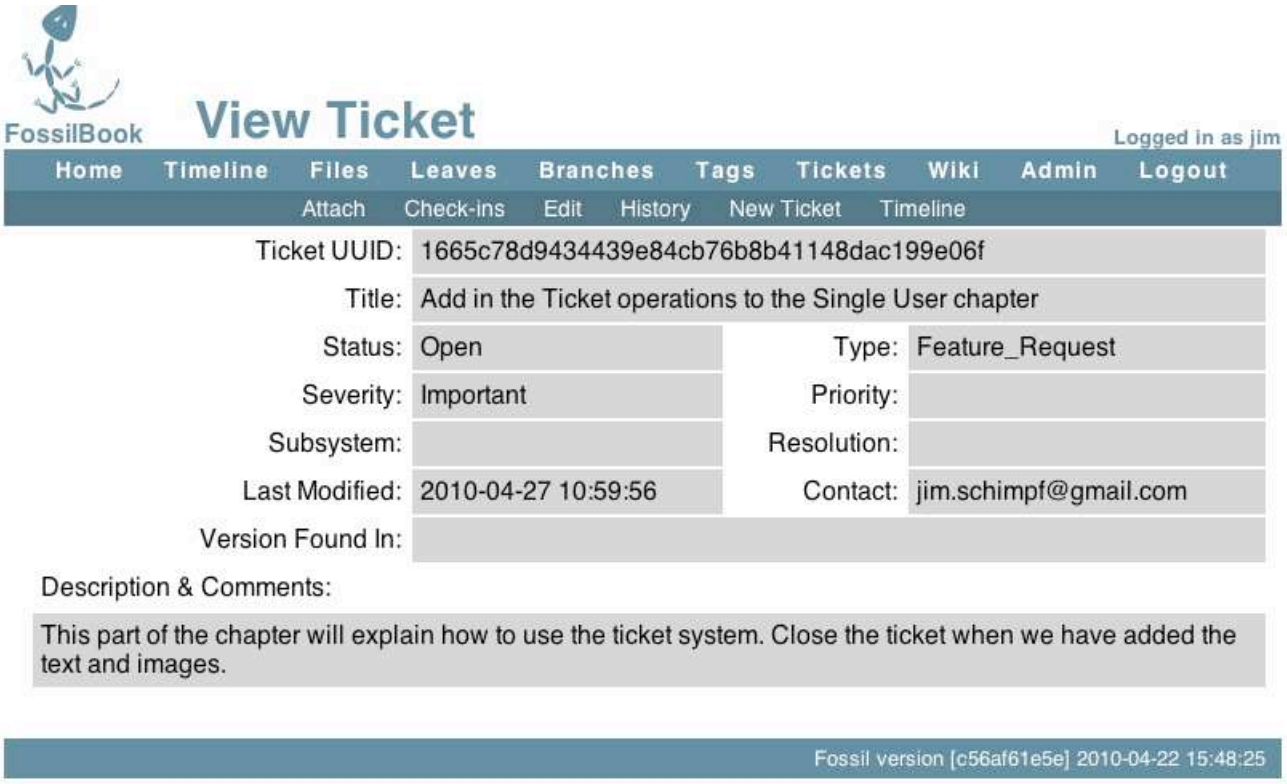

Figure 16: Viewing a Ticket

Finally picking Tickets then All Tickets I can see my new ticket in the list marked as Open and in a distinctive color.

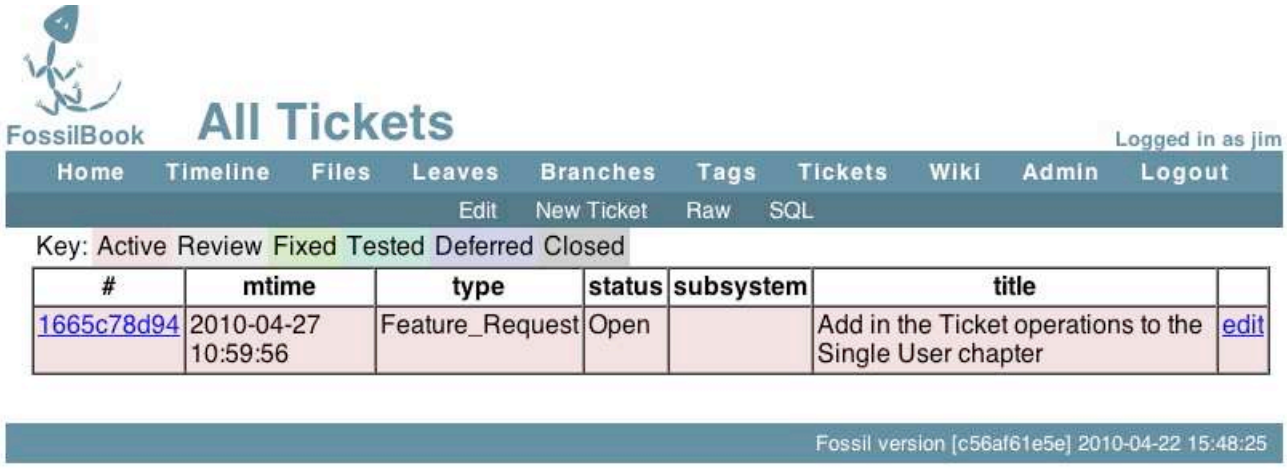

Figure 17: Ticket List with open ticket

I try in handling tickets to have links from ticket to the check-in that addressed the problem and a link from the check-in back to the offending ticket. This way looking at the ticket I can get to the changes made and from the timeline I can get the the ticket and its resolution. To do this I will make sure and put the 10 digit hash label from the ticket into the check-in comment and put a link in the resolved ticket to the check-in.

Since I have now written the chapter and put in all these images of what to do I can now add in all the new images to the repositiory and check this in as another completed operation. And I do that like this:

```
Pandora-2:jschimpf/Public/FossilBook] jim% fossil add Images/Single_user
ADDED  Images/Single_user/config_initial.epsf
ADDED  Images/Single_user/initial_page.epsf
ADDED  Images/Single_user/jim_setup.epsf
ADDED  Images/Single_user/ticket_form.epsf
ADDED  Images/Single_user/ticket_initial.epsf
ADDED  Images/Single_user/ticket_list.epsf
ADDED  Images/Single_user/ticket_submit.epsf
ADDED  Images/Single_user/timeline.epsf
ADDED  Images/Single_user/timeline_detail.epsf
ADDED  Images/Single_user/user_config.epsf
[Pandora-2:jschimpf/Public/FossilBook] jim% fossil commit -m "[1665c78d94] Ticket Use"
```

Figure 18: Ticket resolving check-in

First I added in all the new image files. I am lazy and just told it to add in all the files in the Single_user directory. I have previously added some of those like **config_initial.epsf** but Fossil is smart and knows this and won't add that one twice. Even though it shows it ADDED it really didn't.

The commit line is very important, as you can see I put 10 digit hash code for the ticket in brackets in the comment. As we will see in the Wiki section this is a link to the Ticket, so when viewing the comment in the Timeline or elsewhere you can click the bracketed item and you would go to the ticket.

Now that I have the items checked in I have to close the ticket. I do that by clicking on its link in the ticket list, that will go the the View Ticket window as shown in Figure 17 on the previous page. From there I pick edit and fill it in as shown:

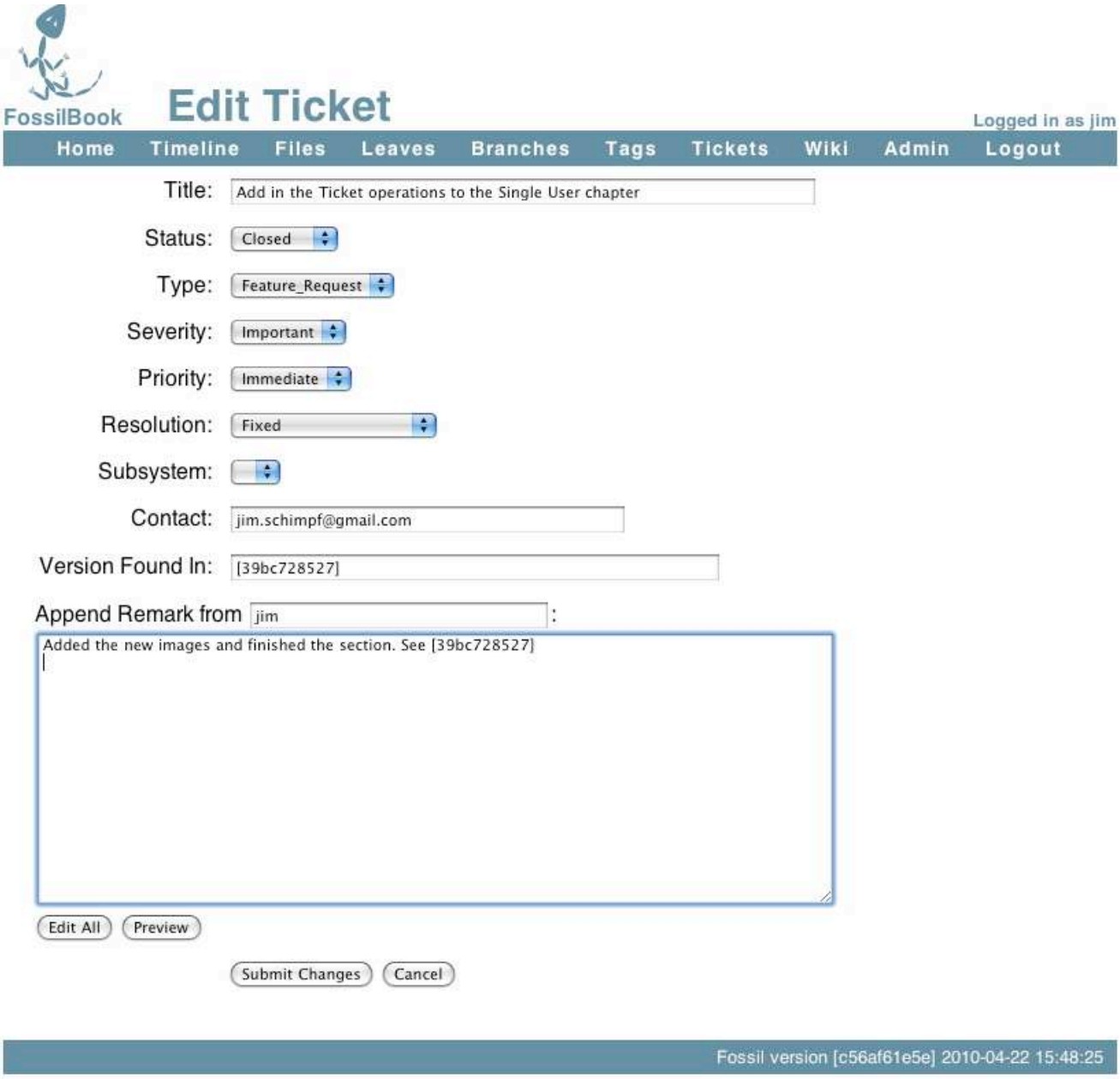

Figure 19: Ticket Finish

I mark it as Closed, if you have code you can mark this as fixed, tested or a number of other choices. Also very important I brought up the Timeline and copied the link for the check-in I had just done in Figure 18 on the preceding page. By doing this my ticket is now cross linked with the check-in and the check-in has a link back to the ticket.

2.5.1 Ticket Summary

- Tickets are a useful way of reminding you what needs done or bugs fixed
- When you commit a change that affects a ticket put the 10 digit hash label of the ticket into the commit comment surrounded by [<10 digit hash>] so you can link to the ticket
- When you close the ticket put in the hash label of the commit that fixed it.

2.6 Wiki Use

As we saw in Figure 5 on page 6 Fossil has a browser based user interface. In addition to the pages that are built in you can add pages to web site. This allows you to add code descriptions, user manuals or other documentation. This keeps all that stuff in one place under version control. A wiki is a web site where you can add pages and links from within your browser. You are given an initial page then if you type [newpage] this text will turn into a link and if clicked will take you to a new blank page. Remember in Figure 6 on page 7 that is the initial page for your project and from there you can add new pages, these are automatically managed by the Fossil version control system, you don't have to add or commit.

Since I did the setup on repository (see Figure 7 on page 7) the home page has changed to this:

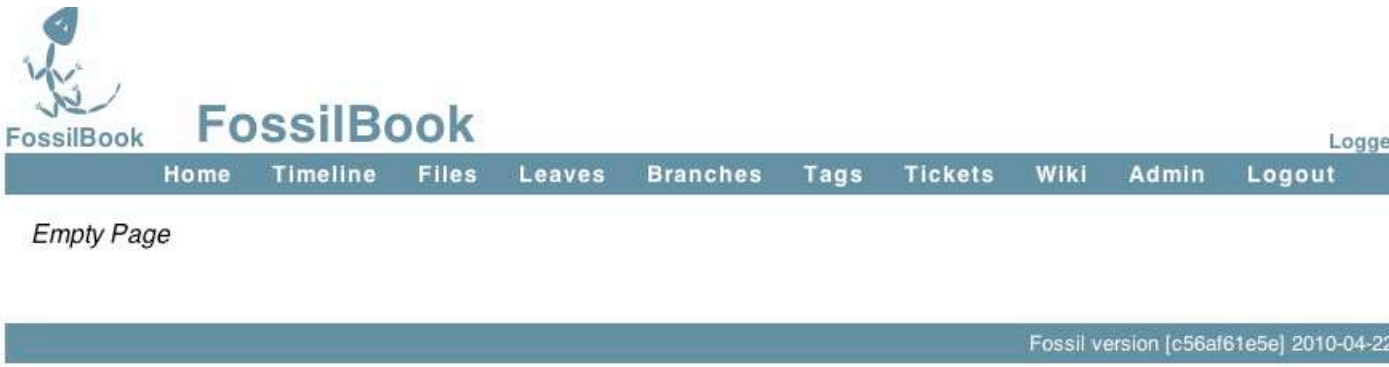

Figure 20: Blank Home Page

Not very helpful so the rest of this chapter I will up use the Wiki functions of Fossil to make this more useful. If I pick the Wiki item from the menu bar I get:

Figure 21: Wiki controls

These are the controls that let you control and modify the wiki. Most important for now is the Formatting rules. This link takes you to a page that describes what you can do to format a wiki page. If you just type text on a page it will appear but be formatted by your browser. You can type HTML commands to control this formating. It's worth your time to carefully read this page and note what you can and cannot do. The page just lists the valid HTML commands, and if you don't know what they mean I would suggest you find a page like this `http://www.webmonkey.com/2010/02/html_cheatsheet/` and keep it handy.

Besides the HTML the most powerful command there is [page] where it links to a new page. This is how you add pages and how you build your web site of documentation for the repository.

Formatting Rule Summary

1. Blank lines are paragraph breaks
2. Bullets are "*" surrounded by two spaces at the beginning of the line.
3. Enumeration items are "#" surrounded by two spaces at the beginning of a line.
4. Indented pargraphs begin with a tab or two spaces.
5. Hyperlinks are contained with square brackets: "[target]" or "[targetlname]".
6. Most ordinary HTML works.
7. <verbatim> and <nowiki>.

We call the first five rules above "wiki" formatting rules. The last two rules are the HTML formatting rule.

Formatting Rule Details

Figure 22: Wiki Formating

I now begin work, what I want to do is change the home page to be non-empty and also put a link on the home page to the PDF of this book. In Figure 21 on the preceding page I click on the first item, FossilBook home page. This takes me to the home page again but now I have an Edit option. We also have a History option so I could look at older versions of the page.

Figure 23: Home page with edit

This shows my initial edit and a preview:

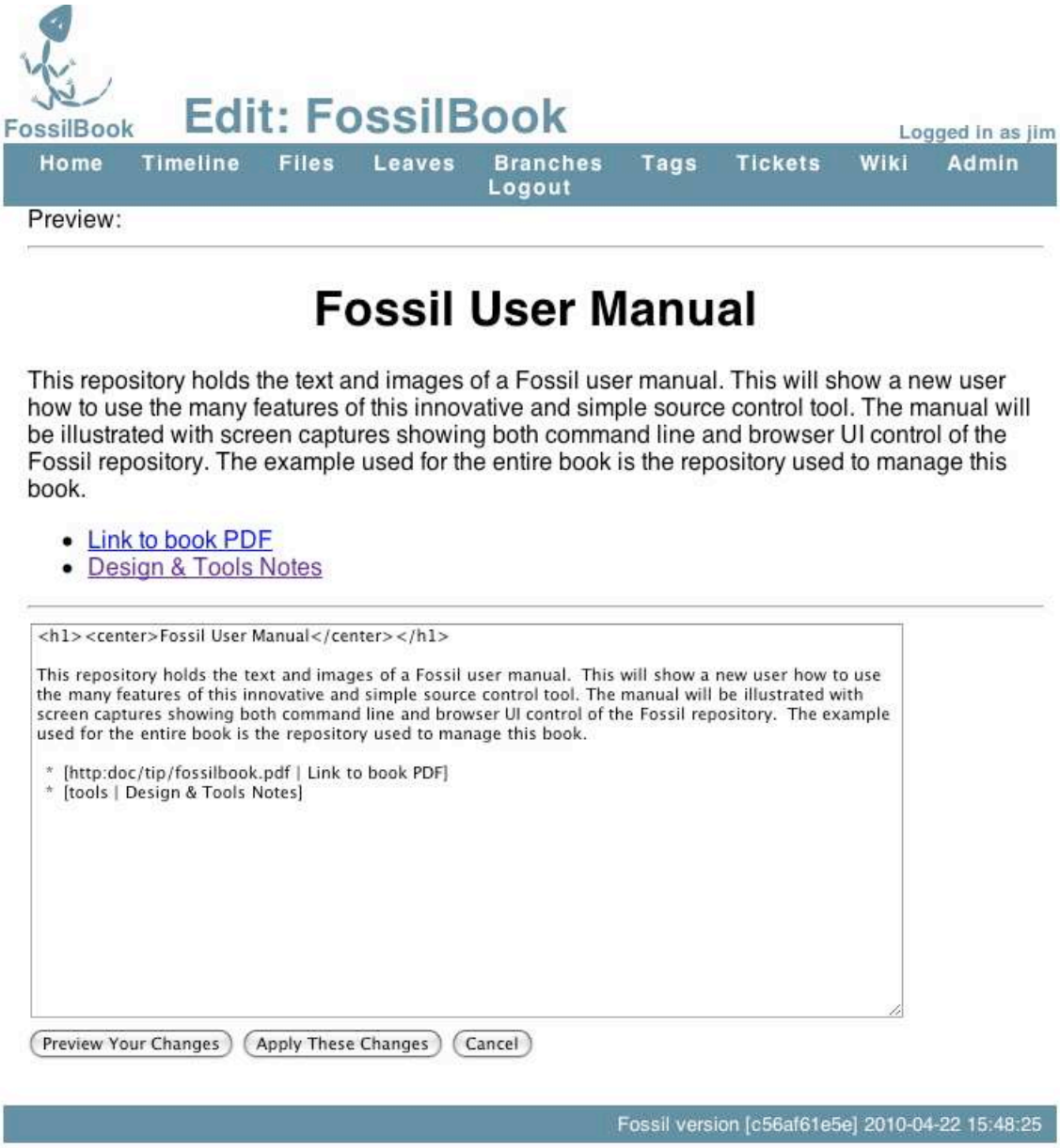

Figure 24: Initial Home page

The bottom section is an edit window where you type things you want displayed and the top is a preview of what the page will be. As you can see I typed some simple HTML to make a large and centered title. The body of the text I just typed and as you see the browser fits the text to the screen. You can have mutiple paragraphs by just putting blank lines between your text. Next I wanted a bullited list and this is done by typing two spaces, a '*' then two more spaces. On each of these lines I have a link to a new (not yet created page). If you look I put these in the form [<new page> | <title>]. This way I can have a long string that describes the link but have a nice short (no embedded

spaces) page name.

One mistake I usually make at this point is to click one of those new links which takes me to a new blank page. OOPS, I have not saved this page yet and I find I've lost my changes on the home page.

OK, I will save my changes and then go to the new pages. I am doing some complicated things here. The first link is to the book PDF, this will be a file I create in the repository. The LyX program I'm using creates the PDF, I will do that save it and add it to the repository. But I don't want to link to a static file, that is I want the most current version of the PDF, the one I save each time I quit for the day. To do this we have to put in a link that looks like this:

```
[http:doc/tip/FossilBook.pdf | Book (pdf) ]
```

This is a special link the Fossil wiki understands, **doc** says this is documentation. **tip** says use the most current version, you could put a version link here. And finally since I am going to put the book PDF at the top level I only need the file name. If it was in a subdirectory I would have to say **doc/tip/subdir/filename.**

The second link is just to a regular page and I can just edit that one like I did this one.

2.6.1 Wiki Summary

- Format your text using HTML commands like <h1>Title</h1> for page headings
- Create and link pages using [<page> | <Link text>]
- The page designation http:doc/tip/<document path relative to repository> will display any document in the repository that your browser can handle (i.e. pdf, http etc)
- Never click on a link till AFTER you have saved the page

References

[1] Mattias Ettrich. Lyx - the document processor.

[2] Dick Grune. Concurrent versions system, a method for independent cooperation. *unpublished*, 1986.

[3] D. Richard Hipp. Fossil home page.

[4] University of Texas. How we use sccs.

[5] Marc J Rochkind. The source control system. *IEEE Transactions on Software Engineering*, SE-1(4):364, December 1975.