title:	Using Revision Libraries in Bazaar
license:	General Public License, V2
copyright:	2004, 2005 Canonical Ltd. ../../bazaar-docs/src/UsingRevisionLibraries.rst:3: (WARNING/2) Cannot extract compound bibliographic field "copyright".
Author:	James Blackwell <jblack@gnuarch.org>
original-location:	jblack@gnuarch.org--2004/bazaar-docs--initial--1.1

Welcome to the next installment of my mini-howtos. In this one, we're going to discuss revision libraries. By the conclusion of this tutorial, you'll have a basic understanding of what revision libraries are, what revision libraries cost you, and most importantly how revision libraries can help you.

I need to dive a little bit into another subject before we go into revision libraries. I assume that by this point, you've managed to get from an existing archive a few times. If so, you've probably noticed that arch builds a working tree all the way from the project's import, or from the last cached revision, whichever comes last. This can take some time to do. What if, after going through all this work, why not keep an copy around in case we need it again? Why, we'd probably store them away in some sort of... "revision library"

Before we go on, lets discuss another problem. Lets say you grab a working copy with "baz get" and make some changes. At this point, two of the things you could do is either perform a "baz changes" or a "baz commit -s'I fixed blah"'. So baz needs to compare the tree you've changed against the way the tree looked before you changed it. In order to do this, baz needs to have an unchanged (a.k.a. "pristine") copy of the revision you got before you started to hack this. So baz, when necessary, actually does the get all over again, storing it off in the {arch} directory somewhere, just so that it can find the changes you made. Rather than do this, wouldn't it be great if arch had already kept a "checked out" working copy at the same time you told it to get? Perhaps in some sort of.... "revision library?"

So its pretty clear how revision libraries can help us. When we set up a revision library, we give baz a place where it can store the results of all of those "baz get"s, so that it can use them again if it needs them. And use them it does! Whenever you type "baz get tuxpucks--devel--1", or you do a "baz changes", arch, instead of building another working copy from scratch, can either refer to the built revision in the library, or just plain make a copy from the library.

How exactly do we set up a revision library? That happens to be an *incredibly* simple task:

```
$ baz my-revision-library /home/jdoe/archives-library
```

From this point forward, baz will automatically add stuff to the library as necessary, and use whats in there automatically. You can actually stop reading now if you want, and live a happy life with baz.

But I know you, gentle reader. You live in a world of supersizing and more, more more. Rather than have you go away half-sated, I'll give you even more. There are two options that you can set for your revision library -- "greedy" and "sparse". These two options, though independant, are complementary.

Lets say that you tell arch to get tux@penguin.org--2004/tuxpucks--devel--1. Arch looks at the archive, and sees that there is base-0 (the start of the version), and 12 patches. Because of the way baz works, it will first get base-0, then get patch-1 and apply it, get patch-2 and apply it, and so forth and so on, until it gets all the way up to patch-12.

Sometimes, baz only temporarily needs a built revision. By default, arch will not add these to the libraries. However, if you like, you can tell arch that *any* time it builds a revision to sock it away by making the library "greedy". You make your revision library greedy in the following way:

```
$ baz library-config --greedy /home/jdoe/archives-library
```

If you want to return baz to the default behavior (nongreedy), then run the following command:

```
$ baz library-config --non-greedy /home/jdoe/archives-library
```

However, when you tell arch to build tuxpucks--devel--1, and it builds all the way up to patch-12, it has to build all of the previous revisions as well. Whether or not baz stores these intermediate built revisions in the library, or throws them away is up to you. If you tell arch to make the library "sparse", then it will only add built revision to the library that you asked baz to make. If you tell arch to make the library "non-sparse", then it will add every revision it built on the way to building the revision you asked for.

If you want arch to only add to the library those revisions that you've asked for, then run the following command:

```
$ baz library-config --sparse /home/jdoe/archives-library
```

If you're more like me, and you say "Disk space is cheap. Store all of them", then you can run the following command instead:

```
$ baz library-config --non-sparse /home/jdoe/archives-library.
```

That leads just to one remaining set of questions. Should your library be greedy? Should it be nonsparse? Should I care? Well, it depends.

If you've got a half-way recent system, then probably you've got more diskspace then you know what to do with. If this is your case, then I suggest you use:

```
$ baz library-config --greedy --non-sparse /home/jdoe/archives-library
```

However, if you're running on some Pentium II with a 64 megs of ram and a 5 gig hard drive, then I recommend the following:

```
baz library-config --greedy --sparse /home/jdoe/archives-library.
```

Welp, its time for me to go again. I appreciate you visiting with me again for another installment of my mini-howto collection.