Here's a quick little example of the DBI arrayref gotcha I'd mentioned on the phone. Code may or may not be correct, since I haven't tested it.

In short, DBI is very memory efficient and returns the same arrayref on the fetchrow_arrayref calls. As always, there's a chance I may be out of date with this, since I haven't double checked in a little while and like all things, it may have gone away.

Let's say you have a small table with some useful values in it. I dunno, a lookup table. Say it's a list of states that you always need in a list of abbreviations. You load it out of the database, but on subsequent hits you can cache it. Here's some code along the lines of what you'd see in the chapter:

```
{
        my @states;
        sub useful_data {
                return @stack if @stack;

                my $sth = $dbh->prepare('select name, abbreviations from states');

                while (my $arrayref = $sth->fetchrow_arrayref) {
                        push @states, $arrayref;
                }

                return @states;

        }
}
```

For simplicity's sake, let's say there are only 4 states = NY, New York; IL, Illinois, TX, Texas; WI, Wisconsin. This code would return @states, but if you dumped out the structure, you'd see:

```
$VAR0 = [
        $VAR0->[3],
        $VAR0->[3],
        $VAR0->[3],
        ['WI', 'Wisconsin']
];
```

This is because you've pushed on 4 copies of the same arrayref and DBI was changing the values each time, so you end up with a whole bunch of pointers to the last arrayref, which contains the last row fetched.

Here's a way around it:

```
{
```

```
        my @states;
        sub useful_data {
                return @stack if @stack;

                my $sth = $dbh->prepare('select name, abbreviations from states');

                while (my $arrayref = $sth->fetchrow_arrayref) {
                        push @states, [ @$arrayref ];
                }

                return @states;

        }
}
```

I bolded the line. All I did was dereference the array and stuff it into a new anonymous array. Problem solved.

Will you hit this case often? Dunno. But the first few times I ran into it I wasted like two days trying to figure out what was going wrong and why in the world I only had the last row of my fetch in my cached array.

-Jim.....