Conclusion

Slide with title

All right, that concludes this talk on app architecture.

What You Learned

We went from a very monolithic architecture that looked like this:

Slide: Bad architecture

To something that is much more modular:

Slide: Modular architecture

This one has many more objects, but they are all simpler because they do just one thing. If the parts are small, then each part is easy to understand and test in isolation.

Each of these sections, which we've called "layers", are mostly separate from each other. The networking layer doesn't know anything about the UI, the domain model doesn't know anything about the networking, and so on. The application layer brings all these pieces together.

You used the principles of decoupling, separation of concerns, and reducing dependencies to make your source code simpler, easier to debug, and easier to maintain.

To be fair, some of this stuff is overkill for many apps. If your app is simple enough, it may be perfectly fine to have the persistence logic in your model objects, for example. Like any other design problem, these things are trade-offs.

But at least you should have some ideas now of how to take all that code out of your view controllers and place it into dedicated objects.

Remember, MVC is not the whole picture. Not everything has to be a model, a view, or a controller. Don't be afraid to break your app into lots of smaller pieces.

If your code becomes hard to understand, or you don't know where to put some new piece of logic, often the solution is to break it up into more objects.

Where To Go From Here?

We didn’t have time to go into every possible architecture issue, so there are some other possibilities left to explore.

Slide with bullet points

For example:

In the New Bid screen the user can type an amount that they want to bid. The validation logic for the amount entered can go into a class of its own. This allows you to reuse it on other screens and in other projects.

You can consider moving the table view data source into a class of its own. For example, if you do that for ActivityViewController, you can also move the currency and date formatters into that new data source class.

An obvious thing that is missing from the current app is a login screen. I left that out for simplicity, but before users can place bids, the server should really know who they are.

The interesting thing about authentication is that it can happen anywhere, at any time, so it’s kind of a “cross-cutting concern”. The ServerAPI code contains some hints of how to implement this, if you’re curious.

Slide with books

If you’re interested in learning more about software architecture, then here are some book recommendations.

- *Domain Driven Design* by Eric Evans
- *Patterns of Enterprise Application Architecture* by Martin Fowler

These books are really about much larger systems than just mobile apps, but they still are worth reading. You can’t really go wrong with these two books.

As far as I know there isn’t a book dedicated to architecture for mobile apps, which is why I’m currently writing one, and it will be published through RayWenderlich.com, so keep an eye on that.

I hope you found this talk useful. If you want to talk more about this stuff, then come see me afterwards. Or you can hit me up on Twitter. My handle is on that slide, @mhollemans.

Unfortunately, I can’t really stick around to talk right now, because I have to give another presentation in the other room right after this one. But after that, feel free to come up to me any time.

Thanks!