Editor's note: this is an older architecture discussion from before we posted them all on the forums right away.

Table of Contents

Debug vs. Release Builds

In the early days, there were small, slow disks, and people saved on disk space and load time by using strip(1) on binaries to remove all the symbols. This is, as far as I can tell, the origin of the debug/release distinction.

Elixir, being by web devs, has web dev concepts; MIXENV supports :dev and :prod, the latter mostly intended to be used with releases. (There's also :test.) Web developers have to deal with such things as "dev" and "production" databases, API secrets, and so forth, and often use these environments to do so.

We don't really have these concerns, and should move away from things like them in some areas; despite the default names Mix uses for configurations, what we actually want is a debug/release build distinction. In particular, despite having a database, we don't have a "production database" in the sense a webapp does, and therefore have no desire to distinguish between a locally spun-up "dev database" vs. a "production database" with customer data.

prod

The default "prod" is our release build, and it's to be considered normal operation. This is the mode used to make releases. Basically, any difference in program behavior from this mode is wrong.

test/dev

I can think of little we would want to guard under the :test environment

specifically, so I’m lumping it in with dev here; there may be
something.

As the “debug build”, and the one developers end up in most often (mix
tasks and iex will assume :dev), these modes are allowed to do things
that don’t change application behavior in any way. Some things they can
do:

- use things like our macros to turn all defp into def and put
docstrings on them, making debugging a bit simpler
- crank up the logging level, resulting in more things dumped to the
screen
- set up iex to be a bit more convenient (i.e., i think this should be
guarded under “not prod”; i have to look into this a bit more)

Some things we should avoid:

- things along the lines of “dev databases”; this should be uniform
among all methods of running * there’s a caveat here. the standard data directory will get
clobbered between dev and prod, and it’s not unreasonable to expect
some random user, using the released software, to download the
source and run it in dev mode to try something out. we don’t want to
clobber their files! while I don’t know precisely how this should be
handled, it shouldn’t be more distinct than having files tagged
“dev” for dev mode.
- there’s a caveat here. the standard data directory will get
clobbered between dev and prod, and it’s not unreasonable to expect
some random user, using the released software, to download the
source and run it in dev mode to try something out. we don’t want to
clobber their files! while I don’t know precisely how this should be
handled, it shouldn’t be more distinct than having files tagged
“dev” for dev mode.
- hardcoded defaults for things configured or configurable in the
release mode. * if there’s useful debugging/test defaults… make a module and put
them in it! then load them from the repl! if we’re not capable of
doing this, change things until we are.
- if there’s useful debugging/test defaults… make a module and put
them in it! then load them from the repl! if we’re not capable of
doing this, change things until we are.
- relatedly to the above, autostarting in any mode is a bit dubious to
me. * startup tasks should be a task given as argument to the release

- opening iex shouldn’t start things up
- saying “iex” isn’t saying “start”
- at any rate, debug/release builds are not where to have this

distinction

- startup tasks should be a task given as argument to the release
- opening iex shouldn’t start things up
- saying “iex” isn’t saying “start”
- at any rate, debug/release builds are not where to have this

distinction

docs

We may want docs to have their own mode; this section is a placeholder until we consider it.

interactive

It’s possible we want interactive use to have its own configuration; I can’t think of when we would.