The thing to think about with feature gating

Ka Wai Cheung, December 2023

I've been thinking about how to create an architecture around feature gating. It's part of some new work I'm beginning for DoneDone. Feature gating is the concept of limiting features in your software product to certain users—for instance, opening them up for more expensive plans while closing them off for cheaper ones.

This seems, at first, like a simple (but not necessarily easy) operation. Create some object knowledgeable of an account's feature allowances and—at least as a primitive approach—splatter conditional statements around such features. If you have the feature you get it; if you don't, you don't. That kind of thing. There are more sophisticated and professional ways of doing this I'm sure.

Initially, it feels like the challenge lives in how you architect the flagging piece. But I've come to find that the real challenge is what you do when someone *had the feature before and now they don't*.

Example: I'm building a new ticket portal feature that will be limited to certain pricing plans. It's an extension to DoneDone's existing mailboxes feature—it gives customers that have previously sent an email to a mailbox the ability to view and reply to their "tickets" via this browser-based ticket portal. This way they don't have to go fishing for the conversation in their own inboxes.

If an account has this ticket portal feature, then they can enable the ticket portal. If they don't have this feature, then they can't. But, what if they had the ticket portal feature for a while, enabled the ticket portal, then downgraded their account so they no longer have the feature? I need to ensure the portal goes away.

My immediate thought is this: Any method related to the ticket portal must check whether the account has this feature available. From methods for setting up and customizing the portal (i.e. the place an account can turn the portal on or off as well as do some custom branding), to the customer ticket listing and detail `GET` methods, to portal-specific authentication methods, to the customer reply `POST` method. I need to sprinkle the check in all of these areas. Those latter customer-centric methods would also have to confirm two things (annoyingly)—both that the ticket portal isn't gated for that account *and* that the account has turned on the ticket portal.

I don't love this approach though. One of those mental sensors that becomes almost automatic the more you code is the one that tells you there's something a bit repetitive going on. Not always in the obvious way like there's an opportunity to DRY up code. Sometimes in a conceptual way.

What gnaws at my senses is this feeling that I'm doing more feature gate checks than I need to do; I *feel* like I should be doing less. A singular linchpin—somehow. Maybe I'm just visualizing the concept of this one large gate around the ticket portal estate and what I've drawn up on the mental whiteboard are a bunch of little gates. Your intuition usually knows something that you'll soon figure out.

Here's what I eventually figure out. The really important part of feature gating is deciding **what feature you are actually gating**. To say that the feature is "the ticket portal" is a broad feature played out in several acts. Instead, I could see the feature as "the ability to turn on the ticket portal." One very small but powerful act.

In other words, I can limit the gate checking to *just* the toggling of the portal. OK, maybe it's a couple checks (there's probably `GET` and `PUT`methods for this) but conceptually it's a check of one sliver of the bigger picture.

If a user can't turn on the ticket portal to begin with, then the rest of the feature gating checks can disappear. The methods to grab ticket data (for a list or an individual detail) and the method that allows a customer to reply to the ticket via the portal only need to check that the portal is turned on—it couldn't be turned on in the first place if "the ability to turn on the ticket portal" isn't granted.

Except, of course, in the case where an account which had enabled the ticket portal now *downgrades* to a plan that doesn't give "the ability to turn on the ticket portal." But then, all I need to do is turn off the ticket portal in the same transaction as the downgrade.

I suppose I should've thought of it this way all along—a feature gate is about deciding what the gate is more than the feature, after all. There are other new features that we'll be gating in our future pricing updates. Not all of them fall in a similar pattern. But, what I'm thinking about for each of them is the linchpin. Where is the singular gate? And what do I need to do to ensure the remaining pieces of the feature go back behind the gate if the keys are handed back?