Which sites use Django?

DjangoSites.org features a constantly growing list of Django-powered sites.

Django appears to be a MVC framework, but you call the Controller the "view", and the View the "template". How come you don't use the standard names?

Well, the standard names are debatable.

In our interpretation of MVC, the "view" describes the data that gets presented to the user. It's not necessarily *how* the data *looks*, but *which* data is presented. The view describes *which data you see*, not *how you see it.* It's a subtle distinction.

So, in our case, a "view" is the Python callback function for a particular URL, because that callback function describes which data is presented.

Furthermore, it's sensible to separate content from presentation – which is where templates come in. In Django, a "view" describes which data is presented, but a view normally delegates to a template, which describes *how* the data is presented.

Where does the "controller" fit in, then? In Django's case, it's probably the framework itself: the machinery that sends a request to the appropriate view, according to the Django URL configuration.

If you're hungry for acronyms, you might say that Django is a "MTV" framework – that is, "model", "template", and "view." That breakdown makes much more sense.

At the end of the day, of course, it comes down to getting stuff done. And, regardless of how things are named, Django gets stuff done in a way that's most logical to us.

<Framework X> does <feature Y> – why doesn't Django?

We're well aware that there are other awesome Web frameworks out there, and we're not averse to borrowing ideas where appropriate. However, Django was developed precisely because we were unhappy with the status quo, so please be aware that "because <Framework X> does it" is not going to be sufficient reason to add a given feature to Django.

Why did you write all of Django from scratch, instead of using other Python libraries?

When Django was originally written a couple of years ago, Adrian and Simon spent quite a bit of time exploring the various Python Web frameworks available.

In our opinion, none of them were completely up to snuff.

We're picky. You might even call us perfectionists. (With deadlines.)

Over time, we stumbled across open-source libraries that did things we'd already implemented. It was reassuring to see other people solving similar problems in similar ways, but it was too late to integrate outside code: We'd already written, tested and implemented our own framework bits in several production settings – and our own code met our needs delightfully.

In most cases, however, we found that existing frameworks/tools inevitably had some sort of fundamental, fatal flaw that made us squeamish. No tool fit our philosophies 100%.

Like we said: We're picky.