Web development starts in the mind. Sure, it may be done in Python, Ruby, C#, Groovy, or any of the other myriad programming languages out there these days, but all of them are just tools. The code for web sites is created in the mind. Conception occurs in the space (or lack of empty space, hopefully) between your ears, and implementation occurs in your programming language of choice.

One of the main goals of this book is to teach you how to architect a site using the Django web framework. But architecting a site, like any piece of software you might be developing, is about a sequence of decisions you need to make for yourself. Some of these decisions are small; others are quite large. A lot of them don’t have an easy answer or one that is immediately obvious.

While I can’t answer your questions for you, and even though my decisions might end up very different from your own, I’m going to talk you through the process. In this way, I hope to show how to translate your train of thought, and the decisions you make along the way, into workable software. And I hope that it makes you some money in the process.

In this book, we’re going to develop an e-commerce application. I chose this kind of web site for a few reasons. First, there is money in selling products online. It’s still very possible for would-be entrepreneurs to conceive of business plans that are financially solvent, solely based on selling stuff to people on the web. For this reason, there is likely a demand for this type of application, and an interest in seeing how it’s done.

Second, I think e-commerce is interesting. An e-commerce project using any particular framework tends to be fairly complex, with lots of ins and outs. They allow you, as a developer, to start by building a simple product catalog, and then go deeper into hooking into third-party payment processors with web service calls. If you can get your head around the logic behind our Django e-commerce project, then there’s probably very little else you won’t be able to figure out how to do with Django. Lastly (and don’t tell anyone this), I’m pretty bad at visual design. I’m a programmer. Visual design for e-commerce tends to be pretty straightforward. Generally, you don’t need to create graphics-heavy grunge designs to wow your audience, like you might need to do for a blog or business-card web site. As far as e-commerce goes: the simpler and cleaner, the better. I can handle that. However, just because we won’t be focusing on design in this book doesn’t mean that it’s not an important part of any e-commerce site. Customers and their purchasing decisions are very much influenced by the look and feel of a web site, so it’s well worth your time to learn a bit more about design and make sure that your site doesn’t turn customers off visually.

So, let’s get right down to it and figure out what our needs are. Maybe you have some idea of what you want to build: I do. Before jumping right in and starting with the coding process, I’m going to take a second to jot down some thoughts about what it is that we’re going to create. It’s common practice for software developers to write a specification for larger projects before they start coding. There are few different kinds of specifications; the one that we’re going to do now is a functional specification, which will describe in plain English what our code is supposed to do.

Specifications vary in style and format. They can be extremely stilted and formal, or they can be much less formal. They can be bulleted lists of features, descriptive text, or take the form of user stories that describe the customer experience on the site. How you choose to write your own specification is not
terribly important, but it is important that you write something, just to make sure that your ducks are in a row and you don’t run into unforeseen problems with your plans down the line. If you make any mistakes or run into problems, it’s much easier (and cheaper) to change a few lines of text on paper than change lots of written code.

I’m going to start by writing a quick, very informal functional specification for our e-commerce project. Take a moment to do the same, and let’s reconvene back here when you’re finished. Done?

Okay, here’s mine: “Modern Musician will be an online e-commerce catalog that sells instruments, sheet music, and music accessories to its customers. We’re going to ship orders from the Cleveland, OH location where our retail shop currently does business.”

Okay, so that’s a little short and doesn’t tell us a whole lot. It doesn’t refine our goals or help us with our decisions or the overall design process. So, I’m going to have a second and much longer go at it:

“Modern Musician will be an online e-commerce catalog that sells instruments, sheet music, and music accessories to its customers. When the user first comes to the site, the home page will display some featured products, a list of categories, and search box that will let them search the catalog. Navigation will be straight across the top below the banner. Product lists should be thumbnails that include a small image of the product, with the product name. Products can be in as many categories as we want. We need to be able to add products and categories through a form so we don’t need to always have computer programmers doing it. These forms need to be protected so only authorized individuals can access them. Products can be added to a cart. Checkout will be a single page where we prompt customers for their information and let them submit the order to us. (How can we inform customers about our privacy policy? Make it always available to them?) We’re going to ship orders from the Cleveland, OH location where our retail shop currently does business. Search Engine Optimization (SEO) should always be a priority. Everything should be stored securely. Our site should be accessible to people with disabilities, such as blind people. How should we order products when several are listed on a single page, such as a category page? We need analytics to track conversions and figure out who’s buying what. The administrative login also needs a place for us to view orders. Orders can only be placed with a valid credit card on the site, and once the card is approved through a real-time lookup, the order can be submitted with a status of “submitted.” Fulfillment will occur on our end, and we’ll set the status of the order to “processed” once they’ve been shipped. We need to handle returns easily. This will require the ability to refund a customer’s money, less any handling fees we might charge them. How can we make this information available to the customer? Where should we explain our return policy, during checkout? A hyperlink to a “Return Policy” page.”

Okay, so that specification was still pretty short and very informal. That’s okay; the point is not to convince venture capitalists to fund us, but to figure out what we’re going to do, internally, by getting the mental juices flowing. It can be free-form. Crawl through the site you’re envisioning in your mind and jot down any thoughts or questions that come to you. Keep it around and add to it as ideas come to you. The more ideas you have in mind from the start, before you start building the thing, the less likely you are to forget something mission-critical and have to go back and redo a bunch of your work.

Selling Stuff Online

I like Shopify. Shopify is an online provider of e-commerce web applications. Sign up with them and you’ll have your own store up online very quickly. It’s very affordable for small businesses, it’s easy to use, and the interface is quite slick. You can even set up a store initially for free, and they’ll still accommodate ten sales transactions on your site per month.

But in this book, I’m going to create an e-commerce application. What we’re going to create in this book is very similar to Shopify. We’re going to create a product catalog, shopping cart, checkout system, and allow users to leave product reviews. Why would you want to roll your own solution from scratch when it’s easy, cheap, and takes so little time to use a provider like Shopify?

The reason has to do with a concept in business referred to as a core competency. Like most terms in the business world, this refers to a concept that seems drop-dead simple and completely obvious to