CHAPTER 5

What Is MVC?

The model-view-controller (MVC) pattern is a software-design pattern used for creating data-driven web applications. A design pattern is a general solution that addresses common software-design challenges. While not a finished design, you may think of a design pattern as a template or set of best practices.

Following the MVC pattern means you intend to keep the presentation layer (view), business logic (controller), and database layer (model) separate. Changes made to one layer will minimally impact the others.

The real benefit of MVC is not seen when writing the code, but rather when maintaining it. Code is in independent units and can be maintained without keeping the entire application in your head.

Team building around MVC is easier. The design lends itself to segmentation among different people or groups. Imagine a View Team that is responsible for great views, a Model Team that knows all about the data, and a Controller Team that understands the application flow and business rules. Each can work on their part of the application concurrently without regard for the other teams. This allows for more rapid application development.

Another great advantage of MVC is code reuse. The application's logic implemented in the model and controller gets reused for each different view.

D. Wolf and A.J. Henley, *Java EE Web Application Primer*,
https://doi.org/10.1007/978-1-4842-3195-1_5

The Model, View, Controller, and Service in Bullhorn

When you think of the model, think of the database. Generally, the model is constructed first. The model must store the data. The model may consist of classes that communicate with the database. The model in Bullhorn is represented by the Oracle database and the entity classes, which represent the tables in Oracle.

Once you create the data model and any classes that are part of the model, move on to the services. The services are all the code that interacts with the model.

Next, move on to the controller. The controller is part of the web application and moves data between the services and the view. The controller also determines which page or servlet is called next. In Bullhorn, the servlets happen to also be the controller. This is not always the case. The controller is simply that code that controls application-specific logic. Since this is a web application, the servlets are in charge of getting data from the view and determining which JSP will display next. If you have Java classes that contain that functionality, they will be part of the controller.

The part of the application the user actually sees is called the *view*. It presents the data to the user and gets data from the user, which is then passed back to the model through the services and controller. The view in Bullhorn consists of JSPs (Java Server Pages) using Bootstrap, CSS (Cascading Style Sheets), JavaScript, and images. All the parts of the view work to create the pages that are displayed in the user's browser. See Figure 5-1.

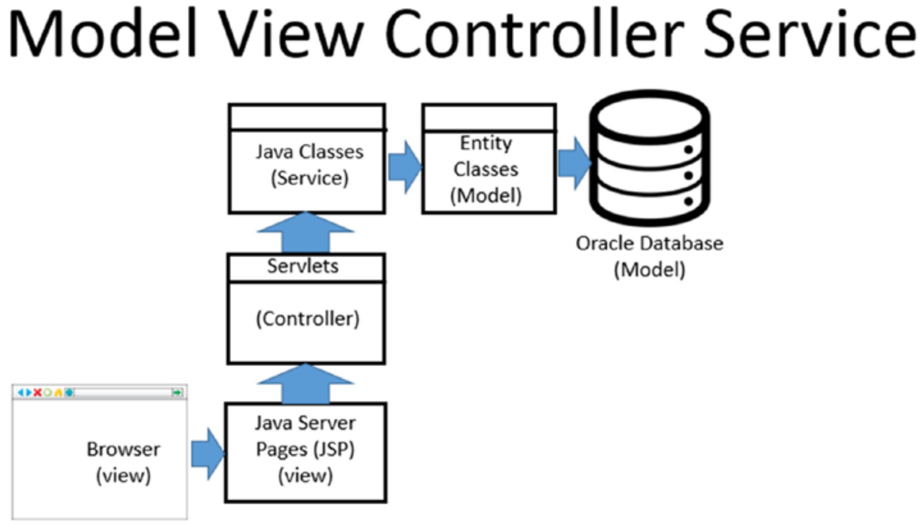

Figure 5-1. *The components of Bullhorn are logically divided into layers called Model, View, Controller, and Service*

Tip Perform validation in every layer. Data can come to your application through ways unanticipated by you when you initially develop it, not just through the browser. For example, it may come to be that you need to import information into your database. Or, you may later write a web service that interacts directly with your service layer.
