Two key features in a web application are user input and state. We need user input to create useful applications. The user selects some function or enters some data, and our application responds accordingly. In a traditional deployed application, this interaction model is intrinsic to the software. In a web application, we rely on HTML forms.

We rely on state in order to store data on behalf of the user and access it in response to web requests. Web application state can be complex, because the underlying technology is broadly stateless. Fortunately, ASP.NET provides a range of different state mechanisms, and you just need to pick the one that best suits your application’s needs. In this chapter, we will look at the features of the four most important and widely used states: form, view, session, and application.

To explore these features, we will walk through producing a simple calculator that solves one of my triathlon training issues. I’m a number-oriented guy, and I record all of my training sessions. When I run and cycle, I use a training computer wristwatch with various sensors that give me a lot of detailed information, including distance, pace, heart rate, and so on. All of this data is uploaded into my training diary.

The sensors communicate with the watch wirelessly, but at a frequency that doesn’t pass very far through water. When I train in the swimming pool, I need to manually record the number of laps I swam and how long it took to swim them. From this, I can estimate the calories I have burned and the distance I swam. Along the way, I need to convert the distance from meters (which is how pool lengths are expressed where I live) to miles (which is how I record my other training). The examples in this chapter are based on a simple conversion calculator that uses my time and distance in meters to calculate the distance in miles, the calories I have burned (an estimate), and the pace at which I swam.

Working with Forms

In this section, we’ll build a web application that is based on a form. HTML forms (which rely on the HTML `<form>` element) are the foundation for web applications. As with our previous examples, we’ll start with the HTML and then add the C# code as a code-behind file. As we create this project, you’ll learn how you can build on the HTML control model that you saw in Chapter 4 to define an HTML form and process the form when it is submitted to the ASP.NET server.

Creating a Form

As with the previous chapters, we need to create an ASP.NET project before we can begin to explore the features that are the focus of this chapter. To create the project, follow these steps:
1. In Visual Studio, select File ➤ New ➤ Project.
2. Click the Visual C# templates in the Installed Templates section, select the Web category, and then click the ASP.NET Empty Web Application template, as shown in Figure 6-1.

![Figure 6-1. Selecting the Empty Web Application template](image)

3. Enter SwimCalculator as the name for the new project, and then click OK to create the project.
4. We want to add a dynamic web page, so select Select Project ➤ Add New Item and choose the Web Form item from the list of templates, as shown in Figure 6-2.