7
SOA and WCF

In the previous chapters, we covered a lot of interesting topics, such as:

- How layered and tiered architectures work
- How to implement an n-tier architecture in our web applications
- How to use the ASP.NET MVC framework to completely de-couple our UI from the code logic and make our GUI more unit-test friendly
- How to use design patterns to write better and more robust code

In this chapter, we will learn about another famous architecture, known as **Service Oriented Architecture** (**SOA**). We will also see how we can implement SOA using the latest Microsoft programming framework for communication between inter-connected systems, known as **Windows Communication Foundation** (**WCF**). This chapter is not intended to be a comprehensive overview of either SOA or WCF, but is intended to give developers a general idea of SOA-based architectures, and where WCF fits in to this.

This chapter covers the following topics:

- Understanding application size, scope and granularity
- What is SOA
- Why we need SOA
- SOA using Web Services
- What is Window Communication Foundation (WCF)
- How we can implement SOA using WCF

Understanding Application Size, Scope, and Granularity

Change is the only thing common to all software projects. No matter how perfect the architecture is, or how robust the code is, we cannot guarantee that the business needs will not change in the future. The core business logic may remain the same, but new "non-core" changes can arise, such as the introduction of new product items, modifications to data display routines, and so on. We cannot avoid change. A good architecture will adapt to change rather than fight it.

Requirement changes and modifications to the code, and their impact on the software application, depend on the actual scope and size of the application itself. So before we go ahead with examining how we can manage changes, we first need to understand how changes relate to the application's size and scope.

Small Applications Versus Big Applications

Web applications (or for that matter any kind of application that we may develop) can broadly be categorized as big or small.

If the application has limited scope in terms of size as well as complexity, then dealing with frequent changes in it, might be manageable. The developer working on such an application can quickly make changes and upload them—a process that will not take much time due to the limited scope and size of these web applications. Dealing with changes in a large application is altogether a different matter.

But how do we define or categorize applications as big or small? Here are some parameters that can help us define the scope of a web application. Applications can be categorized as "small", if they meet the following criteria:

- **Thin business rules:** The application's business logic is not complex, which means that the business layer is either too "thin" or non-existent (the web tier may talk directly with the data layer). For example, a simple website that has only a few tables in the database (possibly used for small forms such as "contact us" submissions), or a simple guestbook, or a simple time tracking application.
- **Limited inter-application communication:** The application does not need to talk with other external third-party applications in the same environment (say, within the same company, or within a network of computers, a LAN, and so on). An application can be labeled as "small", if it is not sharing and is not dependent upon other external applications. Examples of such a "small" application would be an e-card based website, or a small shopping cart in an e-commerce website.