In an n-tier DevForce application, the client tier communicates with an **EntityServer** on a separate application server tier, and the application server in turn communicates with the data tier.

The EntityServer is responsible for accessing data sources and helping to secure your application. The EntityManager in the client application makes requests to the *EntityServer* whenever querying (when the query can't be satisfied from cache) or saving entities, and to call custom service methods.

The *EntityServer* is implemented as several Windows Communication Foundation (WCF) services and can be hosted in different ways, as shown below.

| Deployment | Description |
|---|---|
| Console Application | This is perhaps the simplest deployment, and best suited for development purposes. The *EntityServer* services are h *ServerConsole.exe* application. The services will shutdown when you terminate the console application. |
| Windows Service | The *EntityServer* services may be hosted by a Windows service, installed by the *ServerService.exe* application. Once can start when the machine boots, and can be started and stopped via the Windows Services management console. |
| Internet Information Services (IIS) | The most robust deployment of an *EntityServer* is via IIS. IIS has built-in support for WCF services, and provides r and fault tolerance capabilities. With an IIS deployment, you'll configure a web site with the appropriate configura *EntityServer* will start whenever a client application connects. The *EntityServer* might stop and start with more frec the process. You'll use the IIS Manager to configure and manage the deployment. |
| **Windows Azure** | See our website for more information and a sample deployment. |

We'll discuss the details of each deployment in the following sections.

## Service or Server?

You've probably seen the name **EntityService** used throughout these pages. You've seen it in the <objectServer> element in your config file. You'll see it when you customize WCF configuration with a <*serviceModel*> definition. Why are we using this name, yet talking about the **EntityServer** here? (And for that matter, why did we label that element "objectServer"? Well, that's a story for another day.)

There are several components making up the server. The *EntityService* is a WCF service and functions as little more than a gateway or simple router: its primary purpose is to inform a requesting client which *EntityServer* it will be working with, and to either start that *EntityServer* service or ensure it's already running. It's the *EntityServer* which is responsible for all persistence and security activities.

You might have multiple *EntityServers* too, as one is created to match the specifics of the requesting *EntityManager*. If you are using either a data source extension or custom composition context when you create your *EntityManager*, then it will communicate with an *EntityServer* having those same characteristics. In default configurations DevForce will take care of the details, but this is something to be aware of in more advanced configurations, where you might directly configure the services via either configuration or code.

Here's a sample of the services you might see when using *data source extensions* of "A", "B" and "C".