Regardless of whether your application is a two-tier application performing its own persistence operations, or an n-tier application with a separate application server tier, you might have the need to do some **custom processing** when the singleton *EntityService* starts up and again when it shuts down.

What is the EntityService

We don't often mention the *EntityService* when discussing the application server tier, but you've likely seen the name 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.

The EntityService 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* or ensure it's already running. It's the EntityServer which is responsible for all persistence and security activities. These actions occur whether the application is a standalone application communicating directly with the data tier, or the *EntityServer* has been deployed to an application server tier.

The EntityServiceApplication

The *IdeaBlade.EntityModel.EntityServiceApplication* allows a developer to inject custom logic when the *EntityService* starts, and again at shut down.

Here's a sample implementation:

```
public class CustomEntityServiceApplication : EntityServiceApplication {
  public override void OnServiceStartup(object sender, ServiceStartupEventArgs e) {
    base.OnServiceStartup(sender, e);
     IdeaBlade.Core.TraceFns.WriteLine("My custom EntityServiceApplication is starting.");
   }
  public override void OnServiceShutdown(object sender, EventArgs e) {
     IdeaBlade.Core.TraceFns.WriteLine("My custom EntityServiceApplication is stopping.");
    base.OnServiceShutdown(sender, e);
   }
 }
```

Your custom *EntityServiceApplication* is found during standard discovery.

At startup, your application might need to launch auxiliary "server-side" processes for example, or perform other one-time actions.

At shutdown, the application can run "server-side" clean-up code. For example, it might shut down the auxiliary services it started or send an email alert reporting that the service is coming down.

Note that you cannot use *EntityServiceApplication* to customize the WCF configuration. Instead use the *IdeaBlade.EntityModel.Server.ServiceHostEvents* for this purpose.