Contents

The **ServerService** is provided with the DevForce installation to host the *EntityServer* as a Windows service.

Deploying the *EntityServer* as a Windows service is a good choice in intranet and trusted domain environments.

You'll often find that it's easier to first test with the ServerConsole before deploying the *EntityServer* as a Windows service. The *ServerConsole* configuration and processing are identical to that of the service, yet the console is easier to setup and to diagnose early problems.

Files and assemblies

The following assemblies are always required:

IdeaBlade.Core.dll
IdeaBlade.EntityModel.dll
IdeaBlade.EntityModel.Edm.dll (version 7.2.2 and later: IdeaBlade.EntityModel.Edm.EF5.dll or IdeaBlade.EntityModel.Edm.EF6.dll)
IdeaBlade.EntityModel.Server.dll
IdeaBlade.Linq.dll
IdeaBlade.Validation.dll

You'll also need the .config file:
ServerService.exe.config

If you're deploying a Code First model, you'll also need the following:

EntityFramework.dll
IdeaBlade.Aop.dll
PostSharp.dll

You'll of course also need all of your own "server-side" assemblies. These are the assemblies holding your entity model(s), and any custom extensions you've implemented.

Don't forget **.NET Framework 4.5**.

You may place the DevForce assemblies in the GAC, although there's generally no compelling reason to do so, unless you have a large number of DevForce applications installed and they are all using the same DevForce version.

You do not need to, and in most cases should not, install DevForce to the target machine. Runtime license information is found in your model assembly and not the registry.

Configuration

Did you notice above that the configuration file used here is named ServerService.exe.config? This is because the executable is named ServerService.exe and we're following standard .NET config file naming and discovery conventions. (In Windows Server 2003, the config file should be named ServerService.config.)

The config file will contain most of what's in your app.config file in your client application. It's usually easiest to copy that file and edit as needed.

Database connection information

You'll usually need the <*connectionStrings*> for any databases your application uses. (You can also use <edmKeys> instead of or in addition to *connectionStrings*, but they may be deprecated in the future.)

Remember the credential information in the connection string: if you include a userid and password you may want to think about encrypting the connection strings section if the server on which the *ServerService* is running cannot be secured. If you use

Windows integrated security, rememeber that the service "log on" account will be the account accessing the database. Using a custom account for the service with only the system and database privileges needed is a good option in this case.

If you are using a custom *DataSourceKeyResolver* to dynamically provide connection information, you will not need to define connection information in the config file.

Logging

You can use the <logging> element to set the file name and location of the debug log, or to turn logging off altogether. You also might want to archive log files. Generally other logging attributes are not needed, or used only for debugging purposes.

Logging on the server is usually a good idea, since the diagnostics provided will help during testing and in resolving deployment problems.

A typical config file might contain the following logging information:

```
<logging logFile="log\DebugLog.xml" archiveLogs="true" />
```

... to create a file named DebugLog.xml in the log folder and archive files automatically.

Also see the Log topic for more information. Remember to secure the folder or file from prying eyes.

ObjectServer

You'll generally use the <objectServer> element to configure the service information for the *EntityServer*.

At a minimum you'll have something like the following:

```
<objectServer remoteBaseURL="http://localhost"
          serverPort="9009"
          serviceName="EntityService"
          >
</objectServer>
```

If you've modified any of the defaults for the <serverSettings> element, such as the *allowAnonymousLogin* or *loginManagerRequired* settings, you'll also need to include that information here.

The <*clientSettings*> apply only to the client application.

Note above that we're using port 9009, one you often see in DevForce samples, but there are no DevForce requirements for this port number, and you can pick any free port. You should, however, not change the service name.

Although we show use of the *http* protocol above, you can also use *net.tcp* and *https*. If using *https* you'll need to create an SSL certificate and map it to the port wanted - see this blog for more information.

The Net.TCP Port Sharing Service enables net.tcp ports to be shared across multiple user processes.

As with any DevForce server or client, you can replace the <*objectServer*> configuration with a WCF <*system.serviceModel*> section which gives you complete control over the configuration of communications. See the advanced configuration topic for more information.

Installing the service

You can find the *ServerService.exe* in the *Tools* subfolder under the DevForce installation. The executable does not have static references to DevForce assemblies, and can be used with any version of DevForce 2012.

You must first install the *ServerService* as a Windows service. Do this as follows:

1. Place all required files and assemblies in a folder, along with the ServerService.exe and ServerService.exe.config
2. Launch the Visual Studio command prompt from the Windows Start menu at **Start / Microsoft Visual Studio 2012 / Visual Studio Tools / Visual Studio Command Prompt**.
3. Change to the folder holding your server files and assemblies
4. At the Command Prompt, run the **InstallUtil** tool as follows:
   installutil ServerService.exe
5. After running InstallUtil, the *IdeaBlade DevForce 2012 Entity Service* will be installed as a Windows service. You can now start, stop, and configure this service using the Services plug-in in the Microsoft Management Console (available at Control Panel / Administrative Tools / Services).

Once the service is running, your client applications can then communicate with the *EntityServer*. Remember to make sure that their <*objectServer*> settings match those of the server.

Note that by default the *ServerService* "publishes" its trace messages. See the **logging** topic for more information.

Securing the application

You've hopefully taken steps to secure your application, even if it will be used only within your organization. You've disabled anonymous access, and authorize user actions. Take a look at the additional steps you can also take.

Troubleshooting

- You receive an *AddressAccessDeniedException* telling you that HTTP could not register your URL because you do not have access rights to the namespace. This is caused because the executable is not running with administrator privileges and HTTP addresses are secured resources. You have two options:
    1. Run the *ServerService* with an administrative account.
    2. Run the Netsh command line tool to register the namespace with the account. The steps are as follows:
        1. Open a command prompt using “Run as administrator” and enter:
        2. netsh http add urlacl url=http://+:9009/ user=DOMAIN\USERNAME
           ...where “9009” is the port you are using for the *EntityServer*, and DOMAIN\USERNAME is the system account to be granted access.
    3. Also see http://blogs.msdn.com/drnick/archive/2006/10/16/configuring-http-for-windows-vista.aspx for more information.