There are a number of resources available to help **debug** your application and diagnose the cause of problems when they occur.

- Use *DebugFns.WriteLine* to write your own diagnostic and tracing messages to the DevForce log.
- In Silverlight, implement a custom logger or use the sample trace viewer to capture trace and debug messages.
- Check the debug and trace messages on both client and server! The historical record of your application processing is normally invaluable to IdeaBlade Support, and can offer many helpful clues in diagnosing problems.

We can't emphasize this strongly enough - understanding the trace messages DevForce generates can help you quickly diagnose problems.

- If you're curious about the SQL generated for a query (and if you're not you should be, since the results may surprise you), the *shouldLogSqlQueries* flag on the <logging> element controls whether generated SQL is written to the log:

```
XML  <logging logFile="DevForceDebugLog.xml" shouldLogSqlQueries="true" />
```

- If you're using SQL Server, learn to use **SQL Profiler** to capture and review all server activity.
- EF Prof, an Entity Framework profiler, is an excellent tool for debugging all database activity from your application.
- Take a spin through Intellitrace.
- If you've deployed to Azure and run into problems, both a custom logger and Intellitrace can be helpful.
- Check the **Event Viewer**. The EntityServer writes a few startup messages to the application Event Log, with a source of "DevForce Object Server". Any problems with initializing the default debug log will be written to the Event Log, as well as startup messages from the console and Windows Service servers (ServerConsole.exe and ServerService.exe). These messages indicate how the services were configured, what the endpoint addresses are, and any errors encountered.
- For communication-related problems in n-tier applications, or just to take a peek on what's being sent between the client and server, Fiddler is very helpful.
- For really difficult problems with n-tier communication problems, the Service Trace Viewer comes in handy. You can find the utility in the Windows SDK Tools (the executable name is SvcTraceViewer.exe).

You must first enable diagnostics to use this utility, for example:

```
XML  <configuration>
       <system.diagnostics>
         <sources>
           <source name="System.ServiceModel" switchValue="Information, ActivityTracing">
             <listeners>
               <add name="traceListener" type="System.Diagnostics.XmlWriterTraceListener"
                    initializeData="c:\temp\ServerConsoleTrace.svclog"/>
             </listeners>
           </source>
         </sources>
       </system.diagnostics>
     </configuration>
```

- Enable ASP.NET trace information for an IIS-hosted *EntityServer* to see information on client-server requests.