Contents

DevForce always generates tracing messages as it runs, on both client and server. By default in most environments these messages will be written to a **log** file. You have control over whether a log file will be produced and where, write your own messages, and take control over the logging process.

In an n-tier application the client and server logs contain different messages, since different activities are performed by each. Although the server log will record the activities (such as login and fetch) of the clients connected to it, its log is not a consolidation of all client logs.

The Debug log

By default, DevForce will create a file of trace and debug messages in all environments except Azure, Silverlight and Windows Store*, as long as you specify the *logFile* in the *logging* element of the ideablade.configuration section:

```
<logging logFile="DevForceDebugLog.xml"/>
```

- See the topic on writing a custom logger for how to implement your own logging in these environments.

To disable writing to the log file, set the *logFile* value to an empty string, or remove the *logging* element from the config.

If you don't have an ideablade.configuration section in your config file, or an error prevented it from loading, DevForce will initialize a default configuration. That default will write a file named **DevForceDebugLog.xml** - to a sub-folder for a web application, otherwise to the same folder as the executable. For web applications, DevForce will use the first of three default sub-folders if present: the *app_data\log* folder, the *log* folder, or the *app_data* folder.

To change the location of the log file, just include the full path and file name wanted. For example, to write the log to a folder named "c:\AppLogs" with a name of "SalesAppLog.xml", set logFile="c:\AppLogs\SalesAppLog.xml". If the folder does not exist, or lacks write permissions, the application will run but a file will not be created. If the folder is a sub-folder of the application root folder, use a relative path name, such as "log\MyLog.xml". Note that a tilde in a relative path name is not supported here.

The <logging> element in more detail

Since we've looked at the <logging> element above, let's look at a few other properties you can use to control logging.

```
<logging logFile="DevForceDebugLog.xml" archiveLogs="false" shouldLogSqlQueries="false" />
```

- Set archiveLogs="true" to archive log files. By default this is off, but when on the previous log file will be renamed with a timestamp indicating the time the file was archived.
- Set shouldLogSqlQueries="true" to write the SQL for all entity queries to the log. This can be particularly helpful in debugging.

We describe other settings in the discussion of real-time trace viewing.

Generate custom messages

The *TraceFns* and *DebugFns* classes (in IdeaBlade.Core) allow you to generate trace and debug messages from your code. Both *TraceFns* and *DebugFns* provide identical behavior, except that *TraceFns* calls will always execute, since in Visual Studio .NET projects the conditional TRACE flag is enabled by default for both release and debug builds. For example, you can use *DebugFns.WriteLine* to write diagnostic logging messages wanted only during testing of debug builds, while *TraceFns.WriteLine* can be used for the logging of messages you want in the deployed application.