

eLearning

anytime,



anywhere
education

EliteCertify

LEADING THE WAY TO SUCCESS

70-320 DEVELOPING XML WEB SERVICES AND SERVER
COMPONENTS WITH MICROSOFT VISUAL C#.NET DEMO



© Copyright 1997-2003, EliteCertify.com, All Rights Reserved.

You are creating a .NET Remoting object. You want to add code to the object to log error messages and warning messages. You want the log messages written to both a log file and to the Windows application log.

Which code segment should you use?

- A. `EventLog eventLog = new EventLog("testkobj");
FileStream fileLog = File.Create("testkobj.log");
Trace.WriteLine(eventLog, "sample message");
Trace.WriteLine(fileLog, "sample message");`
- B. `EventLog eventLog = new EventLog("testkobj");
FileStream fileLog = File.Create("testkobj.log");
Trace.Write(eventLog);
Trace.Write(fileLog);
Trace.WriteLine("sample message");`
- C. `Trace.Listeners.Add(new
EventLogTraceListener("testkobj"));
Trace.Listeners.Add(
 new TextFileTraceListener("testkobj.log"));
Trace.WriteLine("sample message");`
- D. `Trace.Listeners.Add(new EventLogTraceListener());
Trace.Listeners.Add(
 new TextFileTraceListener("testkobj.log"));
Trace.WriteLine("sample message");`

Answer: C

Explanation: Listeners direct the tracing output to an appropriate target, such as a log, window, or text file.

An `EventLogTraceListener` redirects output to an event log. A `TextWriterTraceListener` redirects output to an instance of the `TextWriter` class.

We should take care to use the **new `EventLogTraceListener("remobj")`** constructor.

Note: Any listener in the `Listeners` collection gets the same messages from the trace output methods. If we set up two listeners: a `TextWriterTraceListener` and an `EventLogTraceListener`. Each listener receives the same message. The `TextWriterTraceListener` would direct its output to a stream, and the `EventLogTraceListener` would direct its output to an event log.

Reference: Visual Basic and Visual C# Concepts, Trace Listeners
.NET Framework Class Library, `EventLogTraceListener` Class [C#]

Incorrect Answers

- A:** The EventLog object provides interaction with Windows event logs and filestreams enables writing of data to files. However, they are not appropriate for logging warning and error messages.
- B:** The following statements are incorrect.
`Trace.Write(eventLog);`
`Trace.Write(fileLog);`
The correct usage is `Trace.Write(Parameter)`, where Parameter is either an Object or a String that should be written.
- D:** The EventLogTraceListener Constructor() (with no parameter) initializes a new instance of the EventLogTraceListener class without a trace listener.

You create a serviced component named SessionDispenser. This computer is in the EliteCertify.Utilities assembly and is registered in a COM+ server application. SessionDispenser has multiple callers.

You discover that there are logic problems in the Create New Session method. You want to debug any calls to this method.

What should you do?

- A. Open the SessionDispenser solution.
Set a breakpoint on the CreateNewSession method.
Start the debugger.
- B. Attach the debugger to the client process.
Set a breakpoint on the SessionDispenser.CreateNewSession method.
- C. Attach the debugger to the EliteCertify.Utilites.exe process.
Set a breakpoint on the CreateNewSession method.
- D. Attach the debugger to a Dllhost.exe process.
Set a breakpoint on the CreateNewSession method.

Answer: C

Explanation: We should attach the debugger to the COM+ server application, the EliteCertify.Utilities.exe process.

Note: The Visual Studio debugger has the ability to attach to a program that is running in a process outside of Visual Studio. Once you have attached to a program, you can use debugger execution commands, inspect the program state, settings breakpoints, etc.

Reference: Visual Studio, Attaching to a Running Program or Multiple Programs

Incorrect Answers

A: The debugger must be attached to the program that should be debugged.

B: The debugger should be attached to SessionDispenser component, not to the client process.

D: Dllhost.exe can be useful when debugging a Web server.

You create an XML Web service named LatLong that converts street addresses to latitude and longitude coordinates. EliteCertify Inc. charges for this service and allows only existing customers to use the service.

If a customer ID is not passed as part of a SOAP header, you want the service to refuse the request. You want these service refusal messages to be logged to an event log named LatLongLog. You anticipate that there will be a lot of these log entries over time. A string object named refusalMessage contains the message to log.

Which code segment should you use?

- A. `Event log = new EventLog("LatLongLog");
log.WriteEntry(refusalMessage, EventLogEntryType.Error);`
- B. `EventLog log = new EventLog();
log.Source = "LatLongLog";
log.WriteEntry(refusalMessage, EventLogEntryType.Error);`
- C. `if (!EventLog.SourceExists("LatLongSource")) {
 EventLog.CreateEventSource("LatLongSource",
 "LatLongLog");
}`
`EventLog.WriteEntry("LatLongSource",
 refusalMessage, EventLogEntryType.Error);`
- D. `if (!EventLog.SourceExists("LatLongSource")) {
 EventLog.CreateEventSource("LatLongSource",
 "LatLongLog");
}`
`EventLog log = new EventLog("LatLongLog");
log.WriteEntry(refusalMessage, EventLogEntryType.Error);`

Answer: C

Explanation: First we use the SourcesExists method to search the registry for an existing event source. If it does not exist we create a new.

Note: The EventLog.CreateEventSource method establishes an application, using the specified Source, as a valid event source for writing entries to a log on the local computer. This method can also create a new custom log on the local computer.

Reference: .NET Framework Class Library, EventLog Members

Incorrect Answers

A, B, D: We should only create a new event source only if it does not exist.