

ASP / ADO Coding Practices

(Windows 2000 only)

Purpose

The purpose of this document is to describe a set of ASP / ADO coding practices that will help you create web applications that will function better in a shared hosting environment. Following these guidelines should make your site faster and help minimize the occurrence of dreaded RPC errors.

Causes of RPC Errors

First it is important to understand why RPC errors occur. There is no single definitive answer, however, anecdotal evidence suggests that frequent RPC errors may be caused by:

1. Not closing all objects created within a script
2. Use of file-based databases (Access in particular)
3. Using the Application or Session objects to store other objects

Not closing all objects created within a script.

While ASP is supposed to close all objects when a script terminates, the process that handles this action is not infallible. In this case, an ounce of prevention truly is worth a pound of cure.

Use of ODBC databases (Access in particular)

File-based databases, particularly Access, are not well suited for use on a production web site. Even with just a small, lightly used database, problems can arise. Our recommendation is that our customers use SQL Server databases, and that they connect to the using the OLE-DB driver instead of ODBC. Besides being more reliable, the SQL Server OLE-DB driver is also much faster than the ODBC version.

Using the Application or Session objects to store other objects

This one sets off a huge red-flag. Storing objects in the Session or Application objects introduces thorny issues of thread affinity, thread locking, request serialization, and high memory usage. Our recommendation is that these objects never be used to store other objects, particularly ADO objects.

Better Coding Practices

Object Usage

This one is really simple: Don't create objects until they are needed, close objects as soon as you are done with them. And always explicitly use `Server.CreateObject` to create objects.

Header & Footer Scripts

It's a Great Idea to use standard header and footer scripts to contain functions commonly used in your scripts and to gather information needed by all of your scripts. On a site that makes use of a database, migrating the code to create / destroy ADO objects and establish the database connections to subroutine can be especially beneficial as it will help eliminate a great deal of redundant code and it forces you to deal with database access in a more consistent manner across all of your scripts.

Application Object

Information stored in the Application object is usable by all of the scripts in your application, regardless of the current user or session. Using the Application object to store global configuration information (such as database connection strings) is definitely a Good Idea. It is our recommendation that the Application object never be used to store other objects -- there is

always a better solution.

Session Object

The Session object should be used to store data that is specific to the current session or user. When using the Session object to persist information across scripts, be careful to ensure that a user pressing the Back button in their browser will not cause an error. It is our recommendation that the Session object never be used to store other objects -- there is always a better solution.

Versions of Microsoft's Visual InterDev prior to 6.0 set a bad example of Session object utilization, as they used it to store static information about database connections. With 6.0 this has been fixed, database connection information is now placed in the Application object.

Visual InterDev

Visual InterDev is a phenomenal web development tool, in moderation. As an editor and deployment tool, it simply cannot be beat, but it is not a replacement for programming knowledge. The code generated by Visual InterDev, especially pre-6.0 versions, is overly complicated, prone to errors, and difficult to debug. Generating your code the old fashioned way -- by hand -- will result in code that you are able to understand, troubleshoot, and maintain by yourself.

Databases

The three rules of web databases are: SQL Server, SQL Server, and SQL Server. File-based databases such as Access and FoxPro suffer from poor performance and scalability problems. Anecdotal evidence suggests that even light usage of the Access ODBC driver can cause problems.

SQL Server is fast, can support very active sites, and offers increased reliability. By migrating to SQL Server you receive additional benefits such as support for stored procedures, triggers, an OLE-DB driver, and much more. Plus you'll save time by using tools such as Enterprise Manager or Visual InterDev to manipulate your database directly across the Internet instead of having to download it from your site, make the changes, and then upload again.

If you think that SQL Server is overkill for your particular application, you should reconsider.

ADO vs. OLE-DB vs. ODBC

ODBC is a Microsoft standard for accessing databases. It was the first such standard, and dates back to Windows 3.x. OLE-DB is an updated Microsoft standard created for Microsoft's 32-bit platforms. OLE-DB was designed to be faster, more efficient, and most of all more stable than ODBC. ODBC and OLE-DB are both low-level interfaces; a typical application or web developer would not use these APIs directly.

To make OLE-DB easier for developers using high-level languages such as VBScript, ADO was created. ADO provides a simplified mechanism for accessing OLE-DB databases. To allow OLE-DB (and therefore, ADO) applications to work with older databases that have not been updated to the newer OLE-DB standard, such as Access, Microsoft also created the "OLE-DB Provider for ODBC Databases."

If you are using Access, FoxPro, or SQL Server via a System DSN, your database connection goes through the "OLE-DB Provider for ODBC Databases." Accessing a database involves going through four API layers: ADO -> OLE-DB -> OLE-DB Provider for ODBC Databases -> ODBC Driver.

By switching to SQL Server and specifying it's OLE-DB driver, the OLE-DB Provider for ODBC Databases can be eliminated from the process. Now your database queries will go through just three API layers: ADO -> OLE-DB -> OLE-DB Driver.

Using OLE-DB to connect to your SQL Server database is as simple as changing your connection

string to the following:

Provider = **SQLOLEDB**
Server = **SERVER NAME**
User ID = **USERID**
Password = **PASSWORD**
Initial Catalog = **DBNAME**

Replace the **bold** words with the information specific to your account.

Additional Resources

SQL Server Magazine's list of the top MS SQL Server resources on the web:

<http://www.sqlmag.com/Articles/Index.cfm> .

The ASP Today site, run by Wrox Press, provides useful information on using ASP, ADO, and more: www.asptoday.com/