

Improving Performance Using State Management

a solution for ASP.NET developers

The logo for FarPoint Technologies, featuring the word "FarPoint" in a serif font. The letter "o" in "Point" is replaced by a blue, 3D-rendered sphere with a white highlight and a soft blue glow around it.

FarPoint

Introduction

As an ASP.NET developer, you will probably want to maintain state in your pages. State management can be done in many ways. The method you choose can affect performance; therefore, you should understand the various ways you can manage state and choose the appropriate option for your application.

If you are unfamiliar with state management, review the information in “Web Forms State Management” and “Introduction to Web Forms State Management” in the Microsoft Visual Studio .NET documentation.

You can manage session state while using Spread for Web Forms in different ways, including the following options:

- Saving data to the view state, which is client based
- Saving data to the session state, which is server based
- Saving data to an SQL database, which is server based
- Loading data for each page request

The following sections discuss the advantages and disadvantages of these options. You will want to manage data differently depending on factors such as the amount of data with which you are working. You should review this information to determine the best approach for your application.

[The following information excerpts, summarizes, and adds product-specific information to the Visual Studio .NET documentation topic, “State Management Recommendations.”]

Saving Data to the View State

View state provides client-based state management, where data is written into pages in hidden fields.

To save the data to the view state, set the `IsTrackingViewState` property for the active sheet to `True` (`FpSpread1.ActiveSheetView.IsTrackingViewState = True`). This is the default control setting for Spread for Web Forms.

Advantages and Disadvantages

The advantages of using the view state are:

- No server resources are required.
- It requires less coding.
- It requires less database access.
- Spread manages all its data. The user changes are saved automatically to the control's data model.

The disadvantage of using view state is the impact on performance. Because the view state is stored in the page itself, storing large values can cause the page to slow down when users display it and when they post it.

Usage

The default control settings use the view state to save data. This default setting is best for small data models. If you are using larger data sets, you will probably want to use one of the other state management options.

Example

The following Visual Basic .NET sample code illustrates using the view state to save data.

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
    If (Me.IsPostBack) Then Return

    'connect to NWIND MS Access example with OLE DB
    Dim thisConnection As New
        OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
        Data Source=D:\NWIND.MDB")
    'open connection
    thisConnection.Open()
    'create DataSet to contain related data tables, rows,
    ' and columns
    Dim thisDataSet As New DataSet()

    Dim orderAdapter As New OleDbDataAdapter("SELECT
        EmployeeID, LastName, FirstName, Title FROM Employees",
        thisConnection)
        orderAdapter.Fill(thisDataSet, "Employees")

    FpSpread1.ActiveSheetView.IsTrackingViewState = True
    FpSpread1.ActiveSheetView.DataSource = thisDataSet
    FpSpread1.ActiveSheetView.DataMember = "Employees"

    thisConnection.Close()
    thisConnection.Dispose()
End Sub
```

Saving Data to the Session State

Session state provides server-based state management, where data is saved to separate browser sessions for each user.

To save the data to the session state, set the `IsTrackingViewState` property for the active sheet to `True`, as with saving data to the view state. Then handle the session state in the `SaveOrLoadSheetState` event.

Advantages and Disadvantages

The advantages of using the session state are:

- It offers easier implementation.
- It requires less database access.
- It can handle larger data models.
- Because the view state is small, pages load quickly.
- It enhances data durability. Data placed in session-state variables can survive Internet Information Services (IIS) restarts and worker-process restarts without losing session data because the data is stored in another process space.
- It provides platform scalability. Session state can be used in both multi-computer and multi-process configurations.

The disadvantage of using the session state is the impact on performance. Session state variables stay in server memory until they are either removed or replaced, and therefore can degrade server performance. Session state variables containing blocks of information like large datasets can adversely affect Web server performance as server load increases.

Usage

Using the session state is best if the data is too big to save to the view state.

Example

The following Visual Basic .NET sample code illustrates using the session state to save data.

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
    If (Me.IsPostBack) Then Return
```

```
        'connect to NWIND MS Access example with OLE DB
        Dim thisConnection As New
            OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
            Data Source=D:\NWIND.MDB")
        'open connection
        thisConnection.Open()
        'create DataSet to contain related data tables, rows,
        ' and columns
        Dim thisDataSet As New DataSet()
```

```
Dim orderAdapter As New OleDbDataAdapter("SELECT * FROM
    Orders", thisConnection)
    orderAdapter.Fill(thisDataSet, "Orders")
```

```
        FpSpread1.ActiveSheetView.DataSource = thisDataSet
        FpSpread1.ActiveSheetView.DataMember = "Orders"
        FpSpread1.ActiveSheetView.IsTrackingViewState = True
```

```
End Sub
```

```

Private Sub FpSpread1_SaveOrLoadSheetState(ByVal sender As
    Object, ByVal e As FarPoint.Web.Spread.SheetViewStateEventArgs)
    Handles FpSpread1.SaveOrLoadSheetState
    If (e.IsSave) Then
        Session(e.SheetView.SheetName) =
            e.SheetView.SaveViewState()
    Else
        e.SheetView.LoadViewState(Session(e.SheetView.SheetName))
    End If
    e.Handled = True
End Sub

```

Saving Data to an SQL Database

Database state management saves data to a specified database. To save data to an SQL database, you must install SQL State Management, as explained in the example.

Advantages and Disadvantages

The advantages of using a database to maintain state are:

- Databases are typically very secure, requiring rigorous authentication and authorization.
- Databases offer large capacity.
- Database information can be stored as long as you like, and it is not subject to the availability of the Web server.
- Databases include various facilities for maintaining data integrity.
- Data stored in your database is accessible to a wide variety of information-processing tools.
- There are numerous database tools available, offering wide support and custom configurations.

The disadvantages of using a database to maintain state are:

- Using a database to support state management requires more complex hardware and software configurations.
- Using a database can affect performance, for example due to poor construction of the relational data model. Also, large numbers of queries to the database can adversely affect server performance.

Usage

Using an SQL database for state management is best if you are working with large amounts of data, particularly data that needs to be secure and which needs to maintain integrity.

Example

This example describes how to install SQL state management, and provides code for setting up the Spread control.

To install SQL State Management complete the following instructions:

1. Do one of the following:

If you are using the MSDE version of SQL Server that ships with Visual Studio.NET you need to open a command prompt window and navigate to the `Windows\Microsoft.Net\Framework\<Version>` directory. Once there, issue the following command to run the `InstallSqlState.sql` script:

```
OSQL -S localhost -U sa -P <InstallSqlState.sql
```

`OSQL.exe` is a tool that ships with MSDE and SQL Server. It allows you to apply a T-SQL script to a SQL Server.

If you are using SQL Server 7 or SQL Server 2000 you can follow the directions above or you can open the Enterprise Manager, open the `InstallSqlState.sql` script from the `Windows\Microsoft.Net\<Framework Version>` directory, and execute the script from there.

Whichever method you choose, the script will set up an ASPState database in your SQL Server Group Databases.

2. After you have run the script to set up SQL state management, you need to make a change to your project's `web.config` file. Under the session State section, change the Mode setting from its current setting (most likely "InProc") to "SQL Server". Then configure the `sqlConnectionString` to point to the SQL Server where you installed the T-SQL script `InstallSqlState.sql` as follows:

```
sqlConnectionString="data source=127.0.0.1;user id=sa;password=;"
```

Then you need to put the following Visual Basic .NET code in your Spread control's `SaveOrLoadSheetState` event:

```
Private Sub FpSpread1_SaveOrLoadSheetState(ByVal sender As Object,  
    ByVal e As FarPoint.Web.Spread.SheetViewStateEventArgs) Handles  
    FpSpread1.SaveOrLoadSheetState  
    If e.IsSave Then  
        Session("data" & e.Index) = e.SheetView.SaveViewState()  
    Else  
        Dim o As Object = Session("data" & e.Index)  
        If Not o Is Nothing Then  
            e.SheetView.LoadViewState(o)  
        End If  
    End If  
    e.Handled = True  
End Sub
```

The actual state information that you save is written to the `tempdb` database.

Loading Data for Each Page Request

When you load data for each page request, you are not maintaining state, rather, you are re-creating each page as it is requested.

To load data for every page request, set the `IsTrackingViewState` property for the active sheet to `False` (`FpSpread1.ActiveSheetView.IsTrackingViewState = False`).

Advantages and Disadvantages

The advantages of loading data for every page request are:

- No server resources are required.
- Because the view state is small, pages load quickly.

The disadvantages of loading data for every page request are:

- The programmers must code more to handle the UpdateCommand, InsertCommand and DeleteCommand events to update the database. In addition, the programmers might need to set up row, column, or cell styles for each page request.
- This method requires more database access if the Spread control is bound.

Usage

Load data for every page request when there is a large dataset and you must minimize the use of server resources.

Examples

The following samples illustrate different ways to handle loading data for every page request. Example 1 illustrates loading the data. Examples 2 and 3 illustrate loading the data from SQL queries, which limit the data loaded to the data required for the page.

Example 1

The following sample code illustrates loading data for every page request.

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
    'Put user code to initialize the page here
    OleDbDataAdapter1.Fill(DataSet11, "Orders")
    FpSpread1.ActiveSheetView.DataKeyField = "OrderID"
    FpSpread1.ActiveSheetView.IsTrackingViewState = False
    Me.DataBind()
End Sub

Private Sub FpSpread1_UpdateCommand(ByVal sender As Object,
    ByVal e As FarPoint.Web.Spread.SpreadCommandEventArgs) Handles
    FpSpread1.UpdateCommand
    Dim conn As New OleDb.OleDbConnection()
    conn.ConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0;Password=" & "" & ";User ID=Admin;Data
Source=C:\test\NW" & _
    "ind.mdb;Mode=Share Deny None;Extended Properties=" & "" & ";Jet
OLEDB:System database=" & "" & " & _
    ";Jet OLEDB:Registry Path=" & "" & ";Jet OLEDB:Database
Password=" & "" & ";Jet OLEDB:Engine Type" & _
    "=4;Jet OLEDB:Database Locking Mode=0;Jet OLEDB:Global Partial
Bulk Ops=2;Jet OLE" & _
    "DB:Global Bulk Transactions=1;Jet OLEDB:New Database
Password=" & "" & ";Jet OLEDB:Creat" & _
```

```

    "e System Database=False;Jet OLEDB:Encrypt Database=False;Jet
OLEDB:Don't Copy Lo" & _
    "cale on Compact=False;Jet OLEDB:Compact Without Replica
Repair=False;Jet OLEDB:S" & _
    "FP=False"
    Dim cmdText As String = "UPDATE Orders SET CustomerID = ?,
EmployeeID = ?, Freight = ?, OrderDate = ?, Req" & _
    "uiredDate = ?, ShipAddress = ?, ShipCity = ?, ShipCountry = ?,
ShipName = ?, Shi" & _
    "ppedDate = ?, ShipPostalCode = ?, ShipRegion = ?, ShipVia = ?
WHERE (OrderID = ?)"

    Dim updateCmd As OleDb.OleDbCommand = New
        OleDb.OleDbCommand(cmdText, conn)

    updateCmd.Parameters.Add(New
        System.Data.OleDb.OleDbParameter("CustomerID",
        System.Data.OleDb.OleDbType.VarWChar, 5, "CustomerID"))
    updateCmd.Parameters.Add(New
        System.Data.OleDb.OleDbParameter("EmployeeID",
        System.Data.OleDb.OleDbType.Integer, 0,
        System.Data.ParameterDirection.Input, False, CType(10, Byte),
        CType(0, Byte), "EmployeeID",
        System.Data.DataRowVersion.Current, Nothing))
    updateCmd.Parameters.Add(New
        System.Data.OleDb.OleDbParameter("Freight",
        System.Data.OleDb.OleDbType.Currency, 0,
        System.Data.ParameterDirection.Input, False, CType(19, Byte),
        CType(0, Byte), "Freight", System.Data.DataRowVersion.Current,
        Nothing))
    updateCmd.Parameters.Add(New
        System.Data.OleDb.OleDbParameter("OrderDate",
        System.Data.OleDb.OleDbType.DBDate, 0, "OrderDate"))
    updateCmd.Parameters.Add(New
        System.Data.OleDb.OleDbParameter("RequiredDate",
        System.Data.OleDb.OleDbType.DBDate, 0, "RequiredDate"))
    updateCmd.Parameters.Add(New
        System.Data.OleDb.OleDbParameter("ShipAddress",
        System.Data.OleDb.OleDbType.VarWChar, 60, "ShipAddress"))
    updateCmd.Parameters.Add(New
        System.Data.OleDb.OleDbParameter("ShipCity",
        System.Data.OleDb.OleDbType.VarWChar, 15, "ShipCity"))
    updateCmd.Parameters.Add(New
        System.Data.OleDb.OleDbParameter("ShipCountry",
        System.Data.OleDb.OleDbType.VarWChar, 15, "ShipCountry"))
    updateCmd.Parameters.Add(New
        System.Data.OleDb.OleDbParameter("ShipName",
        System.Data.OleDb.OleDbType.VarWChar, 40, "ShipName"))

```

```

updateCmd.Parameters.Add(New
    System.Data.OleDb.OleDbParameter("ShippedDate",
    System.Data.OleDb.OleDbType.DBDate, 0, "ShippedDate"))
updateCmd.Parameters.Add(New
    System.Data.OleDb.OleDbParameter("ShipPostalCode",
    System.Data.OleDb.OleDbType.VarWChar, 10, "ShipPostalCode"))
updateCmd.Parameters.Add(New
    System.Data.OleDb.OleDbParameter("ShipRegion",
    System.Data.OleDb.OleDbType.VarWChar, 15, "ShipRegion"))
updateCmd.Parameters.Add(New
    System.Data.OleDb.OleDbParameter("ShipVia",
    System.Data.OleDb.OleDbType.Integer, 0,
    System.Data.ParameterDirection.Input, False, CType(10, Byte),
    CType(0, Byte), "ShipVia", System.Data.DataRowVersion.Current,
    Nothing))
updateCmd.Parameters.Add(New
    System.Data.OleDb.OleDbParameter("OrderID",
    System.Data.OleDb.OleDbType.Integer, 0,
    System.Data.ParameterDirection.Input, False, CType(10, Byte),
    CType(0, Byte), "OrderID",
    System.Data.DataRowVersion.Original, Nothing))

Dim sv As FarPoint.Web.Spread.SheetView = e.SheetView
Dim keyValue As String = sv.GetDataKey(e.CommandArgument)

' find the row
Dim rowFlag As Boolean = False
Dim keyCol As Integer = 4 ' order id
Dim r As Integer
For r = 0 To sv.RowCount - 1
    Dim tmp As String = sv.GetValue(r, 4)
    If (tmp = keyValue) Then
        rowFlag = True
        Exit For
    End If
Next

If Not rowFlag Then
    Return
End If

Dim i As Integer
For i = 0 To sv.ColumnCount - 1
    Dim colName As String = sv.GetColumnLabel(0, i)
    If (Not e.EditValues.Item(i) Is
        FarPoint.Web.Spread.FpSpread.Unchanged) Then
        updateCmd.Parameters(colName).Value = e.EditValues.Item(i)
    ElseIf (OleDbUpdateCommand1.Parameters.Contains(colName))
        Then
            updateCmd.Parameters(colName).Value = sv.GetValue(r, i)

```

```

    End If
Next

Try
    conn.Open()
    i = updateCmd.ExecuteNonQuery()
    conn.Close()
    conn.Dispose()
Catch ex As Exception
    ' Update database failed.
    conn.Close()
    conn.Dispose()
End Try
End Sub

```

Example 2

The following sample code illustrates loading in the records needed each time the page is loaded, greatly enhancing performance. Because this database has a unique sequential numeric ID field, you can use the previous and next buttons or the page numbers to navigate the data.

This example requires more coding, but provides a more efficient application. If you want to allow users to edit data, you will need to use code similar to the code in Example 1 to add the UpdateCommand subroutine to your application.

```

Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
    'Put user code to initialize the page here
    If Not Page.IsPostBack Then
        With FpSpread1.Pager
            'Init the pager to show pager numbers
            .Position =
                FarPoint.Web.Spread.PagerPosition.TopCommandBar
            .Mode = FarPoint.Web.Spread.PagerMode.Both
        End With
    End If
    'Get the top row
    Dim topRow As Integer = FpSpread1.Sheets(0).TopRow
    'Get the data
    SetDataModel(topRow)
End Sub

Public Sub SetDataModel(ByVal topRow As Integer)
    Dim ps As Integer = FpSpread1.Sheets(0).PageSize

    Me.SqlSelectCommand1.CommandText = "SELECT TOP " & ps & "
    CategoryID, Discontinued, ProductID, ProductName, QuantityPerUnit,
    Reorder" & _
    "Level, SupplierID, UnitPrice, UnitsInStock, UnitsOnOrder FROM
    Products" & " Where ProductID >=" & topRow + 1 & " Order by ProductID"

```

```

FpSpread1.Sheets(0).IsTrackingViewState = False
DataSet11.Tables("Products").Clear()
SqlDataAdapter1.Fill(DataSet11)

Dim model As MyModel = New MyModel(DataSet11, "Products")
model.TopRow = topRow

'Set the total row count
Dim dbCmd As Data.SqlClient.SqlCommand = New
    Data.SqlClient.SqlCommand("select count(ProductID) from
    Products", SqlConnection1)
SqlConnection1.Open()
model.RowCount = CType(dbCmd.ExecuteScalar(), Integer)
SqlConnection1.Close()

'Assign the model
FpSpread1.Sheets(0).DataModel = model

End Sub

Private Sub FpSpread1_TopRowChanged(ByVal sender As System.Object,
    ByVal e As FarPoint.Web.Spread.SpreadCommandEventArgs) Handles
    FpSpread1.TopRowChanged
    SetDataModel(e.SheetView.TopRow)
End Sub

End Class

Public Class MyModel
    'Create a custom data model
    Inherits FarPoint.Web.Spread.Model.BaseSheetDataModel

    Private dataset As Data.DataSet = Nothing
    Private datamember As String = String.Empty
    Private trow As Integer = 0
    Private rCount As Integer = 0

    Public Sub New(ByVal ds As Data.DataSet, ByVal dm As String)
        dataset = ds
        datamember = dm
    End Sub

    Public Overrides Function GetValue(ByVal row As Integer,
        ByVal col As Integer) As Object
        'Returns the value for the specified cell
        Dim dt As Data.DataTable = Me.GetDataTable()
        If dt Is Nothing Then
            Return Nothing
        Else
            If row < TopRow Or row >= TopRow + dt.Rows.Count Then

```

```

        Return Nothing
    Else
        Dim r As Integer = row - TopRow
        Return dt.Rows(r).Item(col)
    End If
End If
End Function

Public Overrides Function IsEditable(ByVal row As Integer,
    ByVal col As Integer) As Boolean
    Return True
End Function

Public Function GetDataTable() As Data.DataTable
    'Returns the bound table
    If dataset Is Nothing Then
        Return Nothing
    Else
        If datamember Is Nothing Or datamember = String.Empty Then
            Return dataset.Tables(0)
        Else
            Return dataset.Tables(datamember)
        End If
    End If
End Function

Public Property TopRow() As Integer
    'Set/Returns the top row in Spread
    Get
        Return trow
    End Get
    Set(ByVal Value As Integer)
        trow = Value
    End Set
End Property

Public Overrides Property RowCount() As Integer
    'Set/Returns the row count in Spread
    Get
        Return rCount
    End Get
    Set(ByVal Value As Integer)
        rCount = Value
    End Set
End Property

Public Overrides Property ColumnCount() As Integer
    'Set/Returns the column count in Spread
    Get
        Dim dt As Data.DataTable = GetDataTable()

```

```

        If (dt Is Nothing) Then
            Return 0
        Else
            Return dt.Columns.Count
        End If
    End Get
    Set(ByVal Value As Integer)

        End Set
    End Property
End Class

```

Example 3

The following sample code illustrates loading in the records needed each time the page is loaded, greatly enhancing the performance. If your database does not have a unique sequential numeric ID field, this is the approach to use. Navigation is done using the previous and next buttons and cannot be accomplished using the page number navigation.

This example requires more coding, but provides a more efficient application. If you want to allow users to edit data, you will need to use code similar to the code in Example 1 to add the UpdateCommand subroutine to your application.

```

Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
    topRow = FpSpread1.Sheets(0).TopRow
    SetDataModel(topRow, topRow)
End Sub

Private Sub FpSpread1_TopRowChanged(ByVal sender As Object, ByVal e
    As FarPoint.Web.Spread.SpreadCommandEventArgs) Handles
    FpSpread1.TopRowChanged
    SetDataModel(topRow, e.SheetView.TopRow)
End Sub

Public Sub SetDataModel(ByVal oldTopRow As Integer, ByVal newTopRow
    As Integer)
    Dim firstOrderID As Integer = -1
    Dim lastOrderID As Integer = -1
    If Not ViewState("lastOrderID") Is Nothing Then
        lastOrderID = ViewState("lastOrderID")
    End If
    If Not ViewState("firstOrderID") Is Nothing Then
        firstOrderID = ViewState("firstOrderID")
    End If
    Dim ps As Integer = FpSpread1.Sheets(0).PageSize

    If newTopRow > oldTopRow Then
        'Next button

```

```

        Me.SqlSelectCommand1.CommandText = "SELECT Top " & ps & "
CustomerID, EmployeeID, Freight, OrderDate, OrderID, RequiredDate,
ShipAdd" & _
        "ress, ShipCity, ShipCountry, ShipName, ShippedDate,
ShipPostalCode, ShipRegion, " & _
        "ShipVia FROM Orders" & " Where OrderID >" & lastOrderID
& " Order by OrderID"
        ElseIf newTopRow = oldTopRow Then
            'First time in
            Me.SqlSelectCommand1.CommandText = "SELECT Top " & ps & "
CustomerID, EmployeeID, Freight, OrderDate, OrderID, RequiredDate,
ShipAdd" & _
            "ress, ShipCity, ShipCountry, ShipName, ShippedDate,
ShipPostalCode, ShipRegion, " & _
            "ShipVia FROM Orders" & " Where OrderID >=" & firstOrderID
& " Order by OrderID"
        Else
            'Previous button
            Me.SqlSelectCommand1.CommandText = "SELECT Top " & ps & "
CustomerID, EmployeeID, Freight, OrderDate, OrderID, RequiredDate,
ShipAdd" & _
            "ress, ShipCity, ShipCountry, ShipName, ShippedDate,
ShipPostalCode, ShipRegion, " & _
            "ShipVia FROM Orders" & " Where OrderID <" & firstOrderID
& " Order by OrderID DESC"
        End If

        FpSpread1.Sheets(0).IsTrackingViewState = False
        DataSet11.Tables("Orders").Clear()

        If newTopRow < oldTopRow Then
            ' Reverse the order
            Dim tmpTable As Data.DataTable =
                DataSet11.Tables("Orders").Clone()
            SqlDataAdapter1.Fill(tmpTable)

            Dim dr As Data.DataRow
            Dim i As Integer
            For i = 0 To tmpTable.Rows.Count - 1
                dr = tmpTable.Rows(tmpTable.Rows.Count - 1 - i)
                DataSet11.Tables("Orders").ImportRow(dr)
            Next
        Else
            SqlDataAdapter1.Fill(DataSet11)
        End If
        Dim model As MyModel = New MyModel(DataSet11, "Orders")
        model.TopRow = newTopRow

```

```

Dim dbCmd As Data.SqlClient.SqlCommand = New
    Data.SqlClient.SqlCommand("select count(*) from orders",
        SqlConnection1)
SqlConnection1.Open()
model.RowCount = CType(dbCmd.ExecuteScalar(), Integer)
SqlConnection1.Close()

FpSpread1.Sheets(0).DataModel = model

ViewState("firstOrderID") =
    DataSet11.Tables("Orders").Rows(0).Item("OrderID")
Dim dtcount As Integer = DataSet11.Tables("Orders").Rows.Count
ViewState("lastOrderID") =
    DataSet11.Tables("Orders").Rows(dtcount - 1).Item("OrderID")
End Sub

```

Conclusion

Visual Studio .NET provides multiple ways to implement state management. After you review the available methods and the preceding information, choose the one that will optimize the performance of your application, and take into account your data requirements.

If you need additional assistance with Spread for Web Forms, contact FarPoint Technologies Technical Support Department.

About FarPoint

FarPoint Technologies, Inc., a privately held company with corporate headquarters located in Morrisville, North Carolina, USA, is a leading developer and publisher of professional components for Windows development. Our award-winning tools benefit leading corporations, software companies, and independent consultants around the world as a cost-effective solution for building distributed enterprise-wide applications for commercial or in-house use.

Please contact us if you have questions or comments.

Web site	http://www.fpoint.com	
Phone	Main:	(919) 460-4551
	Sales:	(800) 645-5913
	Technical Support:	(919) 460-1887
Email	Main:	farpoint@fpoint.com
	Sales:	fpsales@fpoint.com
	Technical Support:	fpsupport@fpoint.com
Fax	(919) 460-7606	

Notices

This paper was developed to assist FarPoint Technologies customers in their use of the Spread for Web Forms product. The information in this document represents the view of FarPoint Technologies, Inc. on the topics discussed, as of the publication's date. The continually changing market conditions and general dynamic nature of the computer business mandates that FarPoint Technologies cannot guarantee the accuracy of any information published after the date of publication. This paper was prepared solely for informational purposes. FarPoint Technologies makes no warranties, expressed or implied, in this document.

Information in this document is subject to change without notice and does not represent a commitment on the part of FarPoint Technologies, Inc.

© 2003 FarPoint Technologies, Inc. All rights reserved.

Some material © 2003 Microsoft Corporation.

Unless otherwise noted, all names of companies, products, street addresses, and persons contained herein are part of a completely fictitious scenario or scenarios and are designed solely to document the use of a FarPoint Technologies, Inc., product.

Spread for Web Forms is a trademark of FarPoint Technologies, Inc.

ActiveX, Microsoft, Visual Basic, Visual C#, and Visual Studio are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other brand and product names are trademarks or registered trademarks of their respective holders.