Teamwork Network
Wrapping Up

You've just completed the core functionality of the Teamwork Network application. In this project, you'll add on some of the extra features, including a comprehensive search utility and a home page for the user. The search utility can be used to search the data in the entire system to find information for the user. The home page will show the user any new information in the system since the user's last visit. You'll also learn how you can modify this application to add extra features for your own enterprise.

Building the Home Page

When a user logs in to the system, the first page that appears is the customized home page that you'll be building in this project. This page, shown in Figure 1, summarizes all the new information in the system since the user's last visit. It also provides quick links to the subsystems of the application.

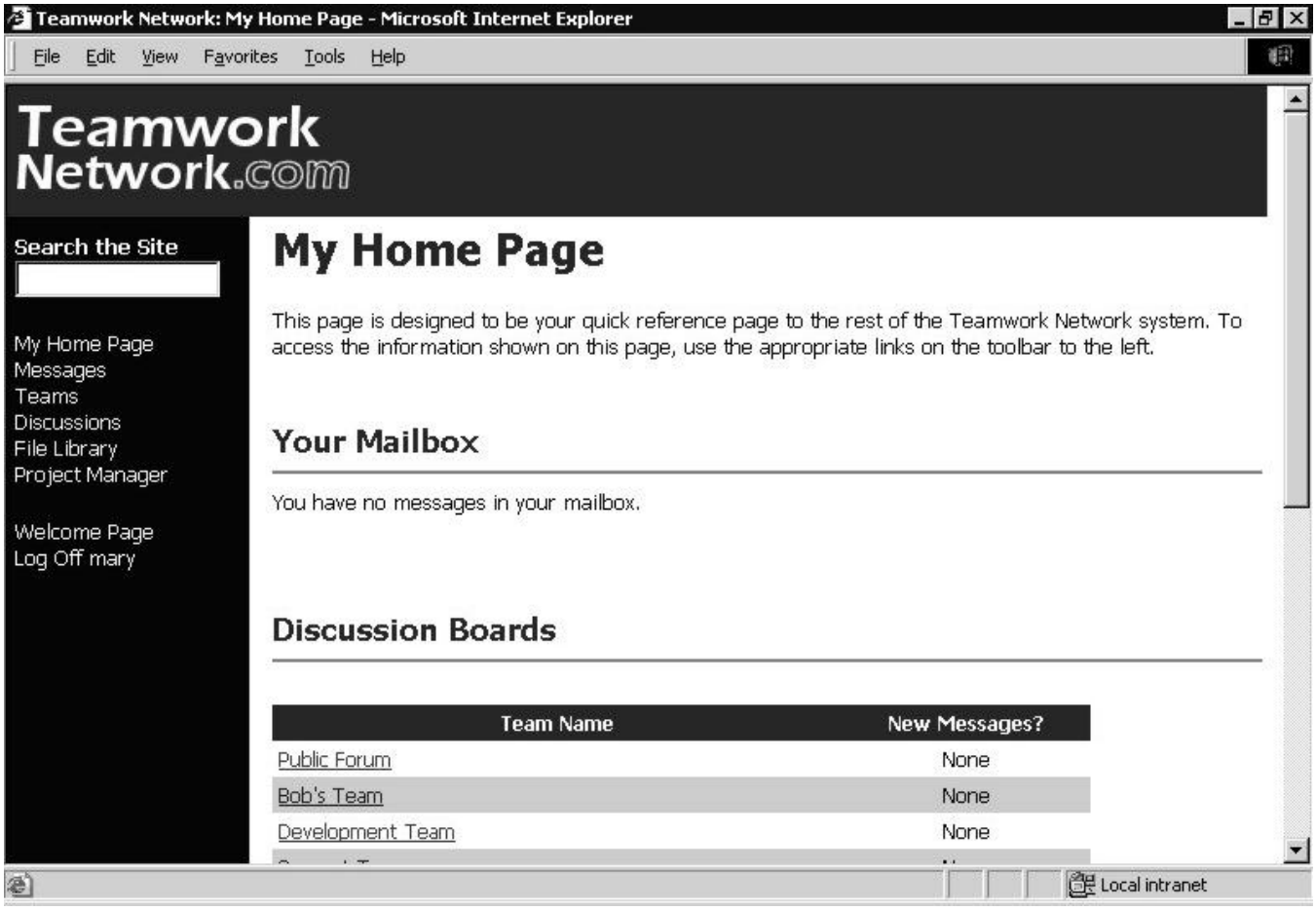

Figure 1 The home page of Teamwork Network.

As you built the other subsystems, you added code to record the user's last visit to certain pages. For instance, when the user went to the messaging subsystem, you recorded the time and date of that visit. That information is used on the home page to determine whether messages in the system are "old" or "new." The same logic is used to determine if new files or discussion board postings were added in any of this member's teams. By summarizing this information and showing it to the user immediately, you save the user from having to dig through all the pages of files, messages, and discussion boards to see the new content.

You might have noticed that the project-tracking system hasn't been mentioned. This is because the information in this system isn't as easy to view in this format. You could, for instance, display when timecards had been added to a project. You could also show when new projects were added to a team. For team managers, you could display when tasks were late or when projects were over budget. However, showing all this information would greatly expand the complexity of this page, which would also increase the time it takes to display it. Most project managers, when using this or similar systems, will have a fairly good idea of the status of each project, so showing the information on the home page becomes less critical.

Building the Stored Procedures

There are a number of stored procedures that we need to build to make this page function properly. These can be created using SQL Server Enterprise Manager, Query Analyzer, or any compatible third-party tool. The first stored procedure retrieves the number of messages received by a member since his or her last visit to the site. You don't show the messages here; instead, you provide a link to the message-viewing page you built in Project 7. The stored procedure is shown in Listing 1.

```
CREATE PROCEDURE dbo.sp_RetrieveMessageCountByMember
@MemberID int
AS
SELECT NewCount =
   (SELECT COUNT(*) FROM tblMessages
   WHERE fkMessageToMemberID = @MemberID AND MessageRead = 0),
OldCount =
   (SELECT COUNT(*) FROM tblMessages
   WHERE fkMessageToMemberID = @MemberID And MessageRead = 1)
GO
```

Listing 1: sp_RetrieveMessageCountByMember

This stored procedure returns two values. The first value, represented by NewCount, is the number of messages received for this member that are unread. Remember that when you read a message, you call a stored procedure that marks that message as read. The second value, represented by OldCount, is the number of messages for this member that are still in the member's mailbox and that have already been read. This will remind the member that he or she has messages stored in the system.

The next stored procedure is used to determine if any new posts have been written in any of the member's discussion forums. The stored procedure code is shown in Listing 2.

```
CREATE PROCEDURE dbo.sp_RetrieveNewPostsByMember
@MemberID int,
@TeamID int
AS
DECLARE @LastVisit datetime
SELECT @LastVisit = LastVisit
FROM tblMemberHistory
WHERE fkTeamID = @TeamID
AND fkMemberID = @MemberID
AND Subsystem = 'D'

SELECT COUNT(*) AS NewPostCount
FROM tblPosts
WHERE fkTeamID = @TeamID
AND PostDate >= IsNull(@LastVisit, '1-1-1900 12:00 AM')
GO
```

Listing 2: sp_RetrieveNewPostsByMember

This stored procedure accepts a member number and team number, which can be zero, to get information about the public forum. The code first retrieves the date of the member's last visit to the discussion forum for the specified team, and that date is then stored in a local variable called LastVisit. The second part of the stored procedure returns the number of posts written in that particular team discussion board since that date. If the member has never visited the team discussion board before, the LastVisit variable will hold a null value. The second query converts that null into a date/time value that is well before any post could have been written.

The next stored procedure returns all the new files since a member's last visit to a particular team's file directories. The code is shown in Listing 3.

```
CREATE PROCEDURE dbo.sp_RetrieveNewFilesByMember
@MemberID int,
@TeamID int
AS
DECLARE @LastVisit datetime
SELECT @LastVisit = LastVisit
FROM tblMemberHistory
WHERE fkTeamID = @TeamID
AND fkMemberID = @MemberID
AND Subsystem = 'F'

SELECT COUNT(*) AS NewFileCount,
MAX(UploadDate) As LastUploadDate
FROM tblFiles
WHERE fkTeamID = @TeamID
AND UploadDate >= IsNull(@LastVisit, '1-1-1900 12:00 AM')
GO
```

Listing 3: sp_RetrieveNewFilesByMember

With the exception of the table name change, this stored procedure works in a similar way to the previous one used to look at the tblPosts table. The code first retrieves the date of the member's last visit to the file library for the specified team, and that date is stored in a local variable called LastVisit. The second part of the stored procedure returns the number of files posted to that team's file library since that date. Again, if the member has never visited the team file library before, the LastVisit variable would hold a null value, and the second query converts that into a date/time value that is well before any file could have been posted.

Now that the stored procedures are complete, you can build the other code needed to make this page run.

Building the Web Page

Since this page doesn't use any new objects, we skip the normal task of building business objects for use in the page. Instead, we can start coding the Web page immediately. The first file to code is the .aspx file called Homepage.aspx. The code for this page is shown in Listing 4.

```
<%@ Page Inherits="TWNW.HomePage" Src="homepage.aspx.vb" %>
<%@ Register Tagprefix="TWNW" Tagname="Header" Src="Header.ascx" %>
<%@ Register Tagprefix="TWNW" Tagname="Footer" Src="Footer.ascx" %>


<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
     <title>Teamwork Network: My Home Page</title>
   <link href="styles.css" rel="stylesheet" type="text/css">
</head>
```

```
<body leftmargin=0 topmargin=0>
<TWNW:Header id="PageHeader" runat="server" />
<p class="pageheading">My Home Page</p>
<p class="text">
This page is designed to be your quick
reference page to the rest of the Teamwork
Network system. To access the information
shown on this page, use the appropriate
links on the toolbar to the left.
</p>
<br>

<p class="subheading">Your Mailbox
<hr noshade>
<asp:label id="lblMessages" class="text" runat="server" />
</p>
<br><br>

<p class="subheading">Discussion Boards
<hr noshade>
</p>

<table cellpadding="4" cellspacing="0" width="500">
<tr class="tableheading">
<td width="70%">Team Name</td>
<td width="30%">New Messages?</td>
</tr>
<asp:label id="lblDiscussions" class="text" runat="server" />
</table>
<br><br>

<p class="subheading">File Libraries
<hr noshade>
</p>

<table cellpadding="4" cellspacing="0" width="500">
<tr class="tableheading">
<td width="70%">Team Name</td>
<td width="30%">New Files?</td>
</tr>
<asp:label id="lblFiles" class="text" runat="server" />
</table>
<br><br>

<p class="subheading">Your Profile
<hr noshade>
<span class="text">
<a href="member_view.aspx?id=<% = Request.Cookies("mID").Value %>">Click
here</a> to view your profile, or
<a href="member.aspx">click here</a> to modify it.
</span>
</p>
<TWNW:Footer id="PageFooter" runat="server" />
```

```
</body>
</html>
```

Listing 4: Homepage.aspx

This page is broken into several logical sections, with each section holding data about one particular part of the Teamwork Network application. The first portion, following the brief introduction, shows information about the member's pending or stored messages. The ASP Label control will be populated with HTML to display an appropriate message and link to the Messages page. The next section will be a table showing the number of new posts in each of this member's team discussion boards. After another visual break provided by the horizontal rule tag (HR), you'll show the user the new files posted since his or her last visit to the team file libraries. The final part of the page includes links to view and modify this member's profile.

The code for this page is shown in Listing 5.

```
Imports System
Imports System.Data
Imports System.Data.SqlClient
Imports System.Web.UI
Imports System.Web.UI.WebControls
Imports TWNWObjects

Namespace TWNW
   Public Class HomePage
      Inherits Page

      Protected lblMessages As Label
      Protected lblDiscussions As Label
      Protected lblFiles As Label

      Sub Page_Load(objSender As Object, objArgs As EventArgs)
         If (Request.Cookies("mID") Is Nothing) Then
            Response.Redirect("login.aspx?msg=403&rURL=" _
               & Request.ServerVariables("SCRIPT_NAME") _
               & "?" _
               & Request.ServerVariables("QUERY_STRING"))
         End If

         Dim DB As New AtWorkUtilities.Database()
         Dim DB2 As New AtWorkUtilities.Database()
         Dim DR As SqlDataReader
         Dim DR2 As SqlDataReader
         Dim strOutput As New System.Text.StringBuilder()
         Dim strColor As String = "tabletext"
         '
         ' Check for any messages sent to the member
         '
         DR = DB.GetDataReader("sp_RetrieveMessageCountByMember " _
            & Request.Cookies("mID").Value)
         If DR("OldCount") = 0 And DR("NewCount") = 0 Then
            strOutput.Append("You have no messages in your mailbox.")
         Else
            If DR("NewCount") = 1 Then
               strOutput.Append("You have 1 new message ")
            ElseIf DR("NewCount") > 1 Then
```

```
      strOutput.AppendFormat("You have {0} new messages ", _
         DR("NewCount").ToString())
   End If
   If DR("OldCount") = 1 Then
      If DR("NewCount") = 0 Then
         strOutput.Append("You have ")
      Else
         strOutput.Append(" and ")
      End If
      strOutput.Append("1 old message")
   ElseIf DR("OldCount") > 1 Then
      If DR("NewCount") = 0 Then
         strOutput.Append("You have ")
      Else
         strOutput.Append(" and ")
      End If
      strOutput.AppendFormat("{0} old messages ", _
         DR("OldCount"))
   End If
   strOutput.Append(" in your mailbox.<br>")
   strOutput.Append("<a href=""messages.aspx"">Click here</a>" _
      & " to view your mailbox.")
End If
lblMessages.Text = strOutput.ToString()
DR.Close()

strOutput = Nothing
strOutput = New Text.StringBuilder()
'
' Check for any new postings in the member's teams
' or in the public forum.
'
DR = DB.GetDataReader("sp_RetrieveTeamsByMember " _
   & Request.Cookies("mID").Value, True)
strOutput.Append("<tr class=""tabletext"">")
strOutput.Append("<td><a href=""posts_view.aspx?tID=0"">")
strOutput.Append("Public Forum</a></td>")
DR2 = DB2.GetDataReader("sp_RetrieveNewPostsByMember " _
   & Request.Cookies("mID").Value & ", 0")
If DR2("NewPostCount") = 0 Then
   strOutput.Append("<td align=middle>None</td>")
Else
   strOutput.AppendFormat("<td align=middle>{0}</td>", _
      DR2("NewPostCount"))
End If
strOutput.Append("</tr>")
strOutput.Append(Environment.NewLine)
strColor = "tabletext_gray"
DR2.Close()

While DR.Read()
   strOutput.AppendFormat("<tr class=""{0}""><td>", strColor)
   strOutput.AppendFormat("<a href=""posts_view.aspx?tID={0}"">", _
      DR("pkTeamID"))
```

```
   strOutput.AppendFormat("{0}</a></td>", DR("Name"))
   DR2 = DB2.GetDataReader("sp_RetrieveNewPostsByMember " _
      & Request.Cookies("mID").Value & ", " _
      & DR("pkTeamID"))
   If DR2("NewPostCount") = 0 Then
      strOutput.Append("<td align=middle>None</td>")
   Else
      strOutput.AppendFormat("<td align=middle>{0}</td>", _
         DR2("NewPostCount"))
   End If
   strOutput.Append("</tr>")
   If strColor = "tabletext" Then
      strColor = "tabletext_gray"
   Else
      strColor = "tabletext"
   End If
   strOutput.Append(Environment.NewLine)
   DR2.Close()
End While
lblDiscussions.Text = strOutput.ToString()
DR.Close()

'
' Check for any new files in the member's teams
' or in the public file library.
'
strOutput = Nothing
strOutput = New Text.StringBuilder()
DR = DB.GetDataReader("sp_RetrieveTeamsByMember " _
   & Request.Cookies("mID").Value, True)
strOutput.Append("<tr class=""tabletext"">")
strOutput.Append("<td><a href=""folder_view.aspx?tID=0"">")
strOutput.Append("Public File Library</a></td>")
DR2 = DB2.GetDataReader("sp_RetrieveNewFilesByMember " _
   & Request.Cookies("mID").Value & ", 0")
If DR2("NewFileCount") = 0 Then
   strOutput.Append("<td align=middle>None</td>")
Else
   strOutput.AppendFormat("<td align=middle>{0}</td>", _
      DR2("NewFileCount"))
End If
strOutput.Append("</tr>")
strOutput.Append(Environment.NewLine)
strColor = "tabletext_gray"
DR2.Close()

While DR.Read()
   strOutput.AppendFormat("<tr class=""{0}""><td>", strColor)
   strOutput.AppendFormat("<a href=""folder_view.aspx" _
      & "?tID={0}"">", _
      DR("pkTeamID"))
   strOutput.AppendFormat("{0}</a></td>", DR("Name"))
   DR2 = DB2.GetDataReader("sp_RetrieveNewFilesByMember " _
      & Request.Cookies("mID").Value & ", " _
```

```
            & DR("pkTeamID"))
         If DR2("NewFileCount") = 0 Then
            strOutput.Append("<td align=middle>None</td>")
         Else
            strOutput.AppendFormat("<td align=middle>{0}</td>", _
               DR2("NewFileCount"))
         End If
         strOutput.Append("</tr>")
         If strColor = "tabletext" Then
            strColor = "tabletext_gray"
         Else
            strColor = "tabletext"
         End If
         strOutput.Append(Environment.NewLine)
         DR2.Close()
      End While
      lblFiles.Text = strOutput.ToString()
      DR.Close()

   End Sub

End Class
End Namespace
```

Listing 5: Homepage.aspx.vb

This code follows the same basic structure as the .aspx file you just built. The code starts by retrieving information about this member's new and stored messages using the sp_RetrieveMessageCountByMember stored procedure. This stored procedure returns two values: the number of new messages and the number of previously read messages. The code here builds a message in HTML to show the user this information, as well as a link to the View Messages page that you built in Project 7. The conditional code here makes sure that the message displayed to the member is grammatically correct.

The next block of code in this page is used to populate the home page with the number of new discussion board postings added since the member's last visit to each discussion board. The sp_RetrieveNewPostsByMember stored procedure that you wrote earlier in this project takes care of that work and returns values that are used in the table. This code handles both the public discussion forum and the individual team boards. In other words, every member will see at least one board listed in this section on the home page. The HTML generated in this section is placed in the ASP Label control that we added to the .aspx page as a placeholder.

Following the discussion board section, the member will see the new files that have been posted to the team file libraries and the public file library. This code works in a similar fashion to that in the previous section, but uses the sp_RetrieveNewFilesByMember stored procedure instead. The resulting HTML code is stored in another ASP Label control in the .aspx file.

At this point, the home page will display an updated summary of the new content added to the application since the member's last visit. Since the other portions of the application are updating the history tables with the member's actions, we can read that information here to make the page more relevant and current for the user.

Building the Search Page

The last function you need to add is a search function. As the amount of content in the application grows, a good search tool becomes a critical piece of functionality to have working well. While the search function you'll build here is somewhat simple, I'll point out places where you can make it more robust. As you build it, you'll probably come up with other enhancements for this page as well.

First, you have to add full-text indexing to your SQL Server database. This will speed up the searches that you'll be doing in your page. Adding full-text indexing to a table can be done by following these steps:

1. Right-click the table name in SQL Server Enterprise Manager. From the pop-up menu, select Full-Text Index Table. From the menu that appears, select Define Full-Text Indexing on a Table.

2. Using the wizard that appears, select the primary key value of the table you picked and the fields that you want to index.

3. You'll be prompted to put the index data into a full-text catalog, which SQL Server will create automatically.

4. Once the catalog is built, you can choose to repopulate the catalog on a schedule.

For more details on full-text indexing, refer to the SQL Server documentation. You don't use the full-text index directly; instead, SQL Server will use it automatically when we use the LIKE keyword in the SQL statements. The first step in building this page is to create the two new stored procedures that will perform our searches for us.

Building the Stored Procedures

This page requires two new stored procedures. The first will search the discussion board postings, and the second will search the file library. You can also add on code to search the Project Tracker and the Timecard tables, but we won't be doing that here. The first stored procedure is shown in Listing 6.

```
CREATE  PROCEDURE sp_SearchPosts
@MemberID int,
@Keywords varchar(200)
AS
SELECT *
FROM tblPosts
WHERE (MessageText LIKE '%' + @Keywords + '%'
OR Subject LIKE '%' + @Keywords + '%')
AND (fkTeamID IN
   (SELECT fkTeamID FROM tblTeamMembers WHERE fkMemberID = @MemberID)
   OR fkTeamID IS NULL OR fkTeamID = 0)
ORDER BY PostDate DESC
```

Listing 6: sp_SearchPosts

This stored procedure accepts a member ID and a keyword string. The member ID is passed so that we don't return search results that the member isn't allowed to see. The keywords are used in a LIKE query to find any records matching the keywords by checking the text of the message and the subject line. You should note that the percent signs are being wrapped around the keyword text. This allows for partial keyword matching in the fields you search. We also verify that the messages are either in one of this member's team discussion boards or in the public discussion areas.

The second stored procedure follows the same pattern and is shown in Listing 7.

```
CREATE PROCEDURE sp_SearchFiles
@MemberID int,
@Keywords varchar(200)
AS
SELECT *
FROM tblFiles
```

```
WHERE (Description LIKE '%' + @Keywords + '%'
OR Name LIKE '%' + @Keywords + '%')
AND (fkTeamID IN
   (SELECT fkTeamID FROM tblTeamMembers WHERE fkMemberID = @MemberID)
   OR fkTeamID IS NULL OR fkTeamID = 0)
ORDER BY UploadDate DESC
```

Listing 7: sp_SearchFiles

Instead of searching the discussion boards, we search the file library names and descriptions using the same logic for checking team membership.

With the stored procedures completed, the next step is to build the Search Results page.

Building the Search Results Page

The last page you need to build will display the results of a user's search. You might be wondering where the search page is. If you hadn't noticed, it's already located on the toolbar and is linked to Search.aspx. In the code we created for it in Project 7, there is no maximum length in the input box, but our stored procedures have a maximum length of 200 characters. You might want to match up the maximum length of the box with the maximum length allowed by the stored procedures, but that's up to you. The Search Results page looks similar to the one shown in Figure 2.

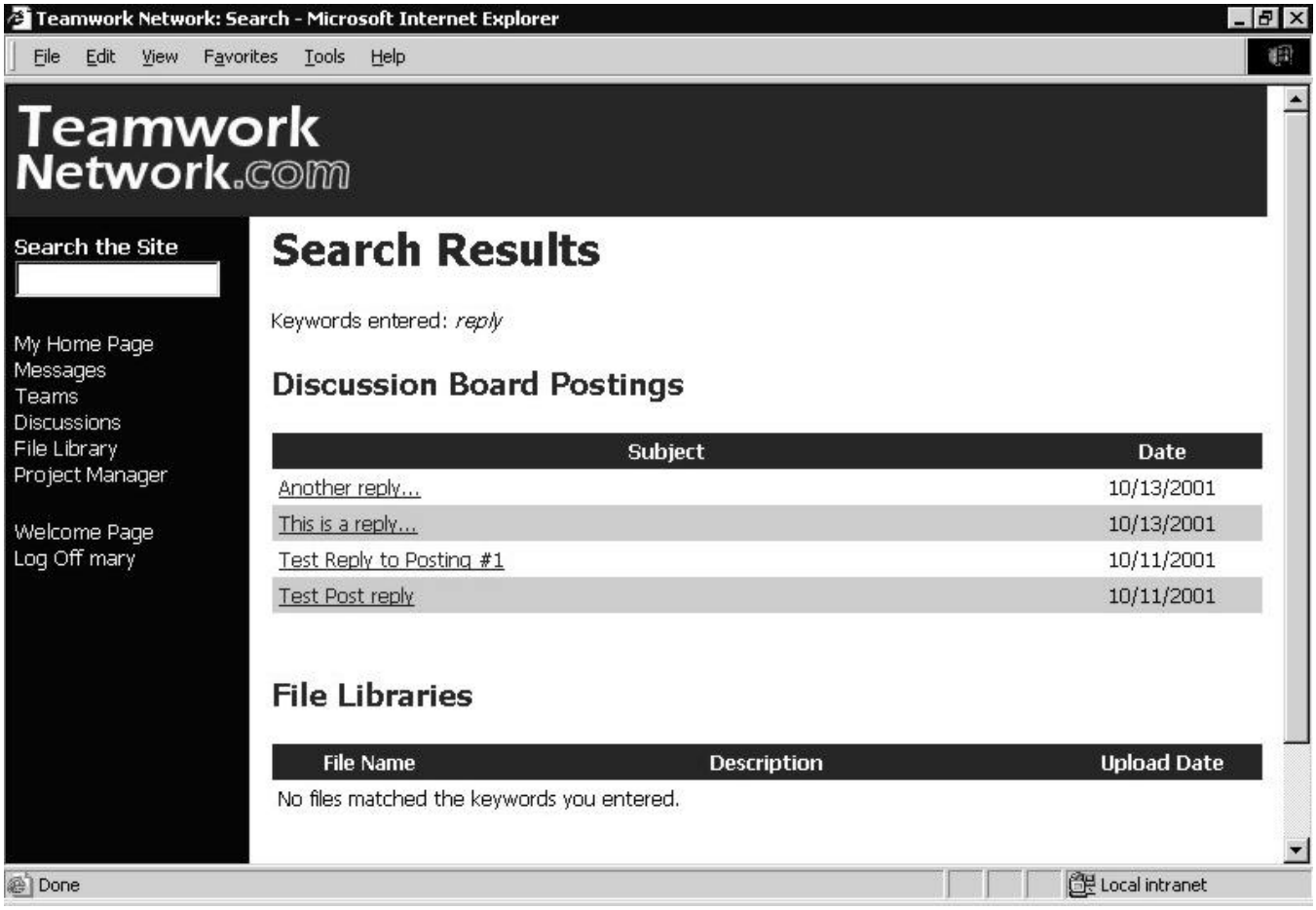

Figure 2 Search Results page.

This page is structured like the home page, with different sections for each type of content being returned. The .aspx file for this page is shown in Listing 8.

```
<%@ Page Inherits="TWNW.Search" Src="search.aspx.vb" %>
<%@ Register Tagprefix="TWNW" Tagname="Header" Src="Header.ascx" %>
<%@ Register Tagprefix="TWNW" Tagname="Footer" Src="Footer.ascx" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
   <title>Teamwork Network: Search</title>
   <link href="styles.css" rel="stylesheet" type="text/css">
</head>
<body>
<TWNW:Header id="PageHeader" runat="server" />
<p><asp:label id="lblPageTitle" class="pageheading" runat="server" /></p>
<p><asp:label id="lblErrorMessage" class="errortext" runat="server" /></p>

<p><asp:label id="lblKeywords" class="text" runat="server" /></p>

<p class="subheading">Discussion Board Postings</p>

<table cellpadding="4" cellspacing="0" width="100%">
<tr class="tableheading">
<td width="80%">Subject</td>
<td width="20%">Date</td>
</tr>
<asp:label id="lblDiscussions" class="text" runat="server" />
</table>
<br><br>

<p class="subheading">File Libraries</p>

<table cellpadding="4" cellspacing="0" width="100%">
<tr class="tableheading">
<td width="20%">File Name</td>
<td width="60%">Description</td>
<td width="20%">Upload Date</td>
</tr>
<asp:label id="lblFiles" class="text" runat="server" />
</table>
<TWNW:Footer id="PageFooter" runat="server" />

</body>
</html>
```

Listing 8 Search.aspx

There are ASP Label controls placed inside each of the result tables, as well as several near the top. Your code will populate those when the search is completed. The code for this page is shown in Listing 9.

```
Imports System
Imports System.Data
Imports System.Data.SqlClient
Imports System.Web
Imports System.Web.UI
Imports System.Web.UI.WebControls
Imports TWNWObjects

Namespace TWNW
```

```
Public Class Search
   Inherits System.Web.UI.Page

   Protected txtSearch As TextBox

   Protected lblPageTitle As Label
   Protected lblErrorMessage As Label
   Protected lblDiscussions As Label
   Protected lblFiles As Label
   Protected lblKeywords as Label

   Sub Page_Load(objSender As Object, objArgs As EventArgs)
      Dim DB As New AtWorkUtilities.Database()
      Dim objCookie As HTTPCookie
      Dim strSearch As String
      Dim DR As SqlDataReader
      Dim strOutput As New System.Text.StringBuilder()
      Dim strColor As String = "tabletext"

      If (Request.Cookies("mID") Is Nothing) Then
         Response.Redirect("login.aspx?msg=403&rURL=" _
            & Request.ServerVariables("SCRIPT_NAME") _
            & "?" _
            & Request.ServerVariables("QUERY_STRING"))
      End If

      objCookie = Request.Cookies("mID")

      lblPageTitle.Text = "Search Results"
      If Request.Form("txtSearch").ToString() <> "" Then
         strSearch = Request.Form("txtSearch").ToString()
         lblKeywords.Text = "Keywords entered: <i>" & strSearch & "</i>"
         DR = DB.GetDataReader("sp_SearchPosts " _
            & Request.Cookies("mID").Value & ", '" _
            & strSearch & "'", True)

         strColor = "tabletext"

         While DR.Read()
            strOutput.AppendFormat("<tr class=""{0}""><td>", strColor)
            strOutput.AppendFormat("<a href=""posts_read.aspx" _
               & "?pID={0}"">", _
               DR("pkPostID"))
            strOutput.AppendFormat("{0}</a></td>", DR("Subject"))
            strOutput.AppendFormat("<td align=center>{0}</td>", _
               DR("PostDate").ToShortDateString())
            strOutput.Append("</tr>")
            If strColor = "tabletext" Then
               strColor = "tabletext_gray"
            Else
               strColor = "tabletext"
            End If
            strOutput.Append(Environment.NewLine)
         End While
```

```
            If strOutput.Length = 0 Then
               lblDiscussions.Text = "<tr><td colspan=3 class=tabletext>" _
                  & "No discussion board " _
                  & "posts matched the keywords you entered.</td></tr>"
            Else
               lblDiscussions.Text = strOutput.ToString()
            End If
            DR.Close()

            DR = DB.GetDataReader("sp_SearchFiles " _
               & Request.Cookies("mID").Value & ", '" _
               & strSearch & "'", True)

            strOutput = Nothing
            strOutput = New Text.StringBuilder()
            strColor = "tabletext"

            While DR.Read()
               strOutput.AppendFormat("<tr class=""{0}""><td>", strColor)
               strOutput.AppendFormat("<a href=""file_download.aspx" _
                  & "?ID={0}"">", _
                  DR("pkFileID"))
               strOutput.AppendFormat("{0}</a></td>", DR("Name"))
               strOutput.AppendFormat("<td>{0}</td>", DR("Description"))
               strOutput.AppendFormat("<td align=center>{0}</td>", _
                  DR("UploadDate").ToShortDateString())
               strOutput.Append("</tr>")
               If strColor = "tabletext" Then
                  strColor = "tabletext_gray"
               Else
                  strColor = "tabletext"
               End If
               strOutput.Append(Environment.NewLine)
            End While
            If strOutput.Length = 0 Then
               lblFiles.Text = "<tr><td colspan=3 class=tabletext>" _
                  & "No files matched the " _
                  & "keywords you entered.</td></tr>"
            Else
               lblFiles.Text = strOutput.ToString()
            End If
         Else
            lblErrorMessage.Text = "Please enter your keywords into " _
               & "the Search box on the toolbar."
            lblDiscussions.Text = "<tr><td colspan=2 class=tabletext>" _
               & "No keywords entered.</td></tr>"
            lblFiles.Text = "<tr><td colspan=2 class=tabletext>No " _
               & "keywords entered.</td></tr>"
         End If
      End Sub
   End Class
End Namespace
```

Listing 9 Search.aspx.vb

The code first determines if any keywords were typed into the Search box. This is checked by looking in the Request.Form collection for a value called txtSearch. If this value is missing, the user didn't type any value into the toolbar box, and an appropriate message is displayed. (The code to display the messages is at the bottom of this file.)

Once the code has determined that keywords were entered, the keywords are displayed at the top of the page and two separate searches are performed: one against the discussion board postings and another against the file library. If you want to search other areas of content, you can just duplicate the format of this code. Any results are shown and linked to the appropriate pages. For postings, the user can click and view the entire thread where the message appears. For files, the user can click and visit the download page. If no results are returned for either type of content, an appropriate message is displayed.

You could modify this tool to restrict the search results based on the type of content, but as a simple search utility, this search runs quickly since it takes advantage of SQL Server full-text indexing. Depending on how much content is in the application, it may be faster to always search all types of content instead of making the user do individual searches. This implementation is up to you to build.

Wrap Up

As you reach the end of the Teamwork Network application and this book, I hope that you've enjoyed building these projects and have come up with some of your own ideas for making them better and more useful. Be sure to visit the live applications running at http://www.10projectswithasp.net and http://www.teamworknetwork.com. You can also post your comments and questions about the book at http://www.10projectswithasp.net in the discussion forums available there.