## Contents

**Preface** ................................................................................................................................. 9
  - Documentation Conventions ................................................................................................. 10
  - Related Publications ............................................................................................................. 11
  - Customer Support .................................................................................................................. 11
  - Information You Should Have ............................................................................................... 11
  - User Feedback ....................................................................................................................... 12
  - Information Builders Consulting and Training ...................................................................... 13

**1. Introducing WebFOCUS Maintain** ...................................................................................... 15
  - Road Map: Where Should You Go? ....................................................................................... 15
  - What Is WebFOCUS Maintain? .............................................................................................. 15
    - Challenge of Accessing Information. .................................................................................. 15
    - How WebFOCUS Maintain Works. .................................................................................... 16
    - N-Tier Applications. ........................................................................................................... 16
    - Leveraging the Power of the WebFOCUS Server. ............................................................. 16
    - What WebFOCUS Maintain Can Do for You. ................................................................... 17
    - What Is Next? ...................................................................................................................... 17
  - Overview of Developing WebFOCUS Maintain Applications ........................................... 17
    - Step 1: Creating the Project. .............................................................................................. 18
    - Step 2: Describing the Data. ............................................................................................ 19
    - Step 3: Logically Partitioning the Application. ................................................................. 20
    - Step 4: Creating the Front End. ......................................................................................... 21
    - Step 5: Creating the Data Access Logic. .......................................................................... 25
      - Procedure: How to Make a Procedure Access a Database. ......................................... 25
      - Procedure: How to Write Maintain Language Code. .................................................... 26
    - Step 6: Setting Up Front End and Data Access Procedures to Call Each Other. ............ 26
      - Procedure: How to Set Up Procedures to Call Each Other. ....................................... 27
    - Step 7: Adding Other Application Components. ............................................................. 27
      - Procedure: How to Use a WebFOCUS Procedure in Your Application. ................... 28
    - Step 8: Testing All the Application Procedures Locally (Optional). ............................... 29
    - Step 9: Setting Up the Deployment Scenario. ................................................................. 30
      - Procedure: How to Set Up Your Deployment Scenario. ............................................. 30
    - Step 10: Deploying and Testing the Application. ............................................................. 32
2. WebFOCUS Maintain Basic Tutorial ........................................... 33
   Before You Begin ........................................................................ 34
   Creating a Project ....................................................................... 34
      Procedure: How to Create the Project Folder. ...................... 35
      Procedure: How to Create the Maintain Procedure. .......... 36
   Familiarizing Yourself With the Explorer .............................. 38
      Procedure: How to View the Files Available to a Project. .... 39
   Adding a Data Source to Your Project ..................................... 40
      Procedure: How to Add a Data Source to Your Project. .... 40
   Familiarizing Yourself With the Project Explorer .................. 41
      Procedure: How to Use a Data Source in a Procedure. .... 41
      Viewing the Structure of a Data Source in the Project Explorer. 42
   Designing a Form ..................................................................... 44
      Procedure: How to Add Data Source Fields to a Form. .... 44
      What Are Data Source Stacks? .............................................. 45
      Procedure: How to Specify a Data Source Stack for Fields on a Form. 45
      Components of a Procedure in the Project Explorer. ........ 46
   Working in the Form Editor .................................................... 47
   Moving Controls on a Form .................................................... 48
      Procedure: How to Move Controls on a Form. ................. 48
   Renaming Field Prompts ........................................................ 49
      Procedure: How to Rename Field Prompts. .................... 50
      Procedure: How to Quickly Rename Field Prompts. .... 52
   Aligning Controls ................................................................... 52
      Procedure: How to Align Entry Fields. ............................ 53
   Saving Your Work .................................................................... 54
      Procedure: How to Save Your Project. .............................. 54
   Selecting a Starting Object .................................................... 55
   Running Your Project Locally ................................................ 56
      Selecting a Starting Object .................................................. 56
      Procedure: How to Select a Starting Object. ................... 56
      Procedure: How to Run Your Project Locally. ............... 56
   Using Radio Buttons .............................................................. 58
      Procedure: How to Add a Group of Radio Buttons to Your Form. 59
      Procedure: How to Add Tool Tip Text. ............................ 61
### 3. WebFOCUS Maintain Advanced Tutorial

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before You Begin</td>
<td>99</td>
</tr>
<tr>
<td>Creating an Update Form</td>
<td>100</td>
</tr>
<tr>
<td>Defining HTML Table Links</td>
<td>101</td>
</tr>
<tr>
<td>Procedure: How to Enable HTML Table Links</td>
<td>102</td>
</tr>
<tr>
<td>Creating the Update/Delete Form</td>
<td>104</td>
</tr>
<tr>
<td>Procedure: How to Add Controls to Update_Form</td>
<td>104</td>
</tr>
<tr>
<td>Copying Controls</td>
<td>106</td>
</tr>
<tr>
<td>Procedure: How to Copy Controls to Update_Form</td>
<td>106</td>
</tr>
<tr>
<td>More About Copying Controls</td>
<td>107</td>
</tr>
<tr>
<td>Updating a Record in a Data Source</td>
<td>107</td>
</tr>
<tr>
<td>Procedure: How to Update a Record in a Data Source</td>
<td>107</td>
</tr>
<tr>
<td>Deleting a Record From a Data Source</td>
<td>110</td>
</tr>
<tr>
<td>Procedure: How to Delete a Record From a Data Source</td>
<td>110</td>
</tr>
<tr>
<td>Deleting a Record From a Data Source Stack</td>
<td>112</td>
</tr>
<tr>
<td>Procedure: How to Delete a Row From a Stack</td>
<td>114</td>
</tr>
<tr>
<td>Using Scripts for User Feedback</td>
<td>117</td>
</tr>
<tr>
<td>Using JavaScript to Confirm a Deletion</td>
<td>118</td>
</tr>
<tr>
<td>Procedure: How to Confirm a Deletion Using JavaScript</td>
<td>118</td>
</tr>
<tr>
<td>Updating a Data Source Using a Read/Write Grid</td>
<td>118</td>
</tr>
<tr>
<td>Procedure: How to Create An Update Form With a Read/Write Grid</td>
<td>119</td>
</tr>
<tr>
<td>Procedure: How to Update Multiple Records Using the Read/Write Grid.</td>
<td>122</td>
</tr>
<tr>
<td>Using a Read/Write Grid</td>
<td>124</td>
</tr>
<tr>
<td>Creating a Welcome Screen Using the Menu Control</td>
<td>124</td>
</tr>
<tr>
<td>Procedure: How to Create a Welcome Screen</td>
<td>124</td>
</tr>
<tr>
<td>Procedure: How to Create a Menu Bar</td>
<td>125</td>
</tr>
<tr>
<td>Assigning Event Handlers to a Menu Item</td>
<td>127</td>
</tr>
<tr>
<td>Procedure: How to Assign Event Handlers to a Menu Item.</td>
<td>128</td>
</tr>
<tr>
<td>Adjusting the Application Flow of Forms</td>
<td>128</td>
</tr>
<tr>
<td>Procedure: How to Adjust the Application Flow of Forms</td>
<td>128</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Adding a WebFOCUS Report to Your Application</td>
<td>129</td>
</tr>
<tr>
<td>Procedure: How to Add a WebFOCUS Report to Your Application</td>
<td>130</td>
</tr>
<tr>
<td>Procedure: How to Define a Stack to Receive WebFOCUS Report Output</td>
<td>130</td>
</tr>
<tr>
<td>Procedure: How to Execute Report Output Into a Stack</td>
<td>133</td>
</tr>
<tr>
<td>Procedure: How to Display Report Output in an HTML Object</td>
<td>133</td>
</tr>
<tr>
<td>Clearing HtmlStack</td>
<td>135</td>
</tr>
<tr>
<td>Procedure: How to Clear HtmlStack</td>
<td>135</td>
</tr>
<tr>
<td>Using a JavaScript Function to Define an Email Link</td>
<td>136</td>
</tr>
<tr>
<td>Procedure: How to Define Email_Script</td>
<td>136</td>
</tr>
<tr>
<td>Adding a Pop-up Calendar</td>
<td>138</td>
</tr>
<tr>
<td>Adding a Pop-up Calendar to Your Forms</td>
<td>138</td>
</tr>
<tr>
<td>Getting Help in the Maintain Development Environment</td>
<td>138</td>
</tr>
<tr>
<td>Procedure: How to View Context-Sensitive Help for a Dialog Box</td>
<td>140</td>
</tr>
<tr>
<td>Procedure: How to View Maintain Language Syntax Help.</td>
<td>140</td>
</tr>
</tbody>
</table>

### 4. WebFOCUS Maintain Concepts 141

<table>
<thead>
<tr>
<th>Set-based Processing</th>
<th>141</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which Processes Are Set-based?</td>
<td>142</td>
</tr>
<tr>
<td>How Does Maintain Process Data in Sets?</td>
<td>144</td>
</tr>
<tr>
<td>Creating and Defining Data Source Stacks: An Overview</td>
<td>144</td>
</tr>
<tr>
<td>Creating a Data Source Stack</td>
<td>145</td>
</tr>
<tr>
<td>Defining Data Source Columns in a Data Source Stack</td>
<td>145</td>
</tr>
<tr>
<td>Creating a Data Source Stack User-defined Columns</td>
<td>149</td>
</tr>
<tr>
<td>Copying Data Into and Out of a Data Source Stack</td>
<td>150</td>
</tr>
<tr>
<td>Referring to Specific Stack Rows Using an Index</td>
<td>151</td>
</tr>
<tr>
<td>Looping Through a Stack</td>
<td>153</td>
</tr>
<tr>
<td>Sorting a Stack</td>
<td>153</td>
</tr>
<tr>
<td>Editing and Viewing Stack Values</td>
<td>153</td>
</tr>
<tr>
<td>Default Data Source Stack: The Current Area</td>
<td>155</td>
</tr>
<tr>
<td>Maximizing Data Source Stack Performance</td>
<td>156</td>
</tr>
<tr>
<td>Controlling a the Flow of a Procedure</td>
<td>156</td>
</tr>
<tr>
<td>Executing Other Maintain Procedures</td>
<td>157</td>
</tr>
<tr>
<td>Calling a Maintain Procedure on a Different Server</td>
<td>158</td>
</tr>
<tr>
<td>Passing Variables Between Procedures</td>
<td>159</td>
</tr>
</tbody>
</table>
Preface

This documentation provides an introduction to WebFOCUS Maintain. It is intended for application developers who are responsible for planning the enterprise software environment and for operating WebFOCUS Maintain.

How This Manual Is Organized

This manual includes the following chapters:

<table>
<thead>
<tr>
<th>Chapter/Appendix</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introducing WebFOCUS Maintain</td>
<td>Familiarizes you with the architecture and features of WebFOCUS Maintain. It also contains a step-by-step overview of developing and deploying applications.</td>
</tr>
<tr>
<td>2 WebFOCUS Maintain Basic Tutorial</td>
<td>Leads you, step-by-step, through the process of creating and running a WebFOCUS Maintain application.</td>
</tr>
<tr>
<td>3 WebFOCUS Maintain Advanced Tutorial</td>
<td>Continues with developing the application introduced in the Basic Tutorial, introducing more advanced topics.</td>
</tr>
<tr>
<td>4 WebFOCUS Maintain Concepts</td>
<td>Contains an overview of WebFOCUS Maintain concepts, including set-based processing, controlling the flow of an application, forms and event-driven processing, reading and writing records, transaction processing, and object-oriented application development.</td>
</tr>
<tr>
<td>A WebFOCUS Maintain Sample Data Sources</td>
<td>Contains the Master Files and structure diagrams of the Maintain-specific data sources.</td>
</tr>
<tr>
<td>B Glossary</td>
<td>Describes key terms found in this manual.</td>
</tr>
</tbody>
</table>
## Documentation Conventions

The following table describes the documentation conventions that are used in this manual.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>THIS TYPEFACE</strong> or <strong>this typeface</strong></td>
<td>Denotes syntax that you must enter exactly as shown.</td>
</tr>
<tr>
<td><strong>this typeface</strong></td>
<td>Represents a placeholder (or variable) in syntax for a value that you or the system must supply.</td>
</tr>
<tr>
<td>underscore</td>
<td>Indicates a default setting.</td>
</tr>
<tr>
<td><strong>this typeface</strong></td>
<td>Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option that you can click or select.</td>
</tr>
<tr>
<td>Key + Key</td>
<td>Indicates keys that you must press simultaneously.</td>
</tr>
<tr>
<td>{  }</td>
<td>Indicates two or three choices. Type one of them, not the braces.</td>
</tr>
<tr>
<td>[  ]</td>
<td>Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis (...).</td>
</tr>
<tr>
<td>.</td>
<td>Indicates that there are (or could be) intervening or additional commands.</td>
</tr>
</tbody>
</table>
Related Publications

Visit our Technical Content Library at [http://documentation.informationbuilders.com](http://documentation.informationbuilders.com). You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have any questions about this product?

Join the Focal Point community. Focal Point is our online developer center and more than a message board. It is an interactive network of more than 3,000 developers from almost every profession and industry, collaborating on solutions and sharing tips and techniques. Access Focal Point at [http://forums.informationbuilders.com/eve/forums](http://forums.informationbuilders.com/eve/forums).

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our website, [http://www.informationbuilders.com](http://www.informationbuilders.com). It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of [www.informationbuilders.com](http://www.informationbuilders.com) also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

Call Information Builders Customer Support Services (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your questions. Information Builders consultants can also give you general guidance regarding product capabilities. Please be ready to provide your six-digit site code number (xxxx.xx) when you call.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Information You Should Have

To help our consultants answer your questions effectively, be prepared to provide the following information when you call:

- Your six-digit site code (xxxx.xx).
- Your WebFOCUS configuration:
  - The front-end software you are using, including vendor and release.
  - The communications protocol (for example, TCP/IP or HLLAPI), including vendor and release.
  - The software release.
Your server version and release. You can find this information using the Version option in the Web Console.

The stored procedure (preferably with line numbers) or SQL statements being used in server access.

The Master File and Access File.

The exact nature of the problem:

- Are the results or the format incorrect? Are the text or calculations missing or misplaced?
- Provide the error message and return code, if applicable.
- Is this related to any other problem?

Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?

What release of the operating system are you using? Has it, your security system, communications protocol, or front-end software changed?

Is this problem reproducible? If so, how?

Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing just the code to access the data source?

Do you have a trace file?

How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

User Feedback

In an effort to produce effective documentation, the Technical Content Management staff welcomes your opinions regarding this document. You can contact us through our website, http://documentation.informationbuilders.com/connections.asp.

Thank you, in advance, for your comments.
Information Builders Consulting and Training

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our website (http://education.informationbuilders.com) or call (800) 969-INFO to speak to an Education Representative.
Chapter 1

Introducing WebFOCUS Maintain

The following topics offer introductory information about WebFOCUS Maintain for new users. These topics also provide an overview of the step-by-step process for developing an application, from creating the interface to deploying the application.

In this chapter:

- Road Map: Where Should You Go?
- What Is WebFOCUS Maintain?
- Overview of Developing WebFOCUS Maintain Applications

Road Map: Where Should You Go?

Welcome to WebFOCUS Maintain.

- If you are new to WebFOCUS Maintain, read What Is WebFOCUS Maintain? on page 15 and then work through WebFOCUS Maintain Basic Tutorial on page 33.
- If you are familiar with the WebFOCUS Maintain 8, read the WebFOCUS New Features manual.
- If you are interested in a quick overview of how to develop a WebFOCUS Maintain application, read Overview of Developing WebFOCUS Maintain Applications on page 17.

What Is WebFOCUS Maintain?

WebFOCUS Maintain is an application development tool that creates web-based data maintenance applications to be deployed across an enterprise (which can be loosely defined as the computer system of a company). Using WebFOCUS Maintain and its multi-platform 4GL language, you can easily create, test, and deploy complex business applications that span the Internet, IBM mainframes, midrange servers, and workstations.

Challenge of Accessing Information

If your company is like most companies, you probably have several different computing systems in operation. For example, your accounting and payroll systems are running on a midrange UNIX computer, while your inventory system is running on an IBM mainframe. Most of the employees have personal computers on their desks and are accustomed to point-and-click interfaces.
What Is WebFOCUS Maintain?

With WebFOCUS Maintain, you can develop applications with graphical front-ends that run on personal computers and access data on any computer system in your company.

**How WebFOCUS Maintain Works**

Using the Form Editor in the Maintain Development Environment (MDE), you develop the application front-end (or user interface). Then, continuing to use the MDE, you develop the code that extracts data from your data sources and updates the data sources with new data. WebFOCUS Maintain applications can also link to existing COBOL, C, or other 3GL programs, CICS transactions, IMS transactions, RDBMS stored procedures, and FOCUS procedures.

At run time, end users start the application and access a WebFOCUS Server where the data or procedures reside. The application extracts data from the data source, displays it for the end users to see, updates the data source with new information, and runs any procedures.

**N-Tier Applications**

WebFOCUS Maintain applications are called *n-tier applications* because they are capable of distributing processing over many platforms. N-tier applications offer the following advantages:

- **You can access data on multiple platforms**, thus forming relationships among disparate data sources.
- **Your application logic runs on the machine most capable of performing it.** For example, PCs are ideally suited for displaying the user interface of your application. On the other hand, MVS machines may not be capable of displaying pretty pictures, but they pack plenty of processing power. A WebFOCUS Maintain application front-end runs on a PC, whereas the data access code can run on an MVS machine.
- **You can speed up your applications.** Procedures that access data can run on the platform where the data resides, thus ensuring that any aggregation or screening takes place immediately. This means that your application is not shipping large quantities of data across a network to be aggregated or screened somewhere else. Less network traffic means increased application speed.

**Leveraging the Power of the WebFOCUS Server**

The WebFOCUS Server is available for every major operating system, MVS, UNIX, Windows, Open VMS, CICS, and VM/CMS. Using its data adapters, WebFOCUS can access every major database management system, including Informix, Sybase, DB2, and Ingres. The WebFOCUS Server can also run procedures written in another language. For example, COBOL, C, or other 3GL programs, CICS transactions, IMS transactions, RDBMS stored procedures, and FOCUS procedures.
Because Information Builders has already worked out the complications of different operating systems, communications protocols, and data access languages, the developer or end user does not need to worry about or even know where their data is coming from.

**What WebFOCUS Maintain Can Do for You**

Using WebFOCUS Maintain, you can:

- Easily develop web-based data maintenance applications with no prior knowledge of HTML, Java®, or complex 3GLs.
- Access and update data on every major operating system and every major database system (using the power of the WebFOCUS Server).
- Access and update data from different platforms at the same time. For example, the inventory system on MVS and the accounting system on UNIX.
- Preserve your investments in existing software, since WebFOCUS Maintain can easily run COBOL, C, or other 3GL programs, CICS transactions, IMS transactions, RDBMS stored procedures, FOCUS procedures, and many more.
- Take advantage of the strengths of your computing systems while sidestepping the weaknesses by partitioning your application among the platforms that can support them.
- Integrate seamlessly with Java, Java applets, JavaScript®, and VBScript.

**What Is Next?**

For more information on how to use WebFOCUS Maintain, work through *WebFOCUS Maintain Basic Tutorial* on page 33.

If you want to develop your own WebFOCUS Maintain applications immediately, read *Overview of Developing WebFOCUS Maintain Applications* on page 17.

**Overview of Developing WebFOCUS Maintain Applications**

Now that you know what a WebFOCUS Maintain application is and how it works, you are ready for a step-by-step view of the development process. This section summarizes the steps.

Another way to get started immediately developing WebFOCUS Maintain applications is by using Update Assist. All you need is a Master File for the data source for the application you want to create. For more information, see the *Developing Reporting Applications* manual.
Step 1: Creating the Project

Your first step is to create the project file for the application. The project file keeps track of all the pieces of the application: procedures, forms, images, scripts, data source descriptions, and so on.

1. From the Windows Start menu, click Programs, WebFOCUS 8.1 Developer Studio, and then WebFOCUS Developer Studio.
   
   WebFOCUS Developer Studio opens.

2. Click the Open MAINTAIN button on the toolbar to open the WebFOCUS Maintain Development Environment.

3. Ensure that Projects on localhost is selected in the Explorer window, and in the File menu, choose New Project.
   
   or

   Right-click Projects on localhost, and click New Project.

   The Create a Project - Step 1 of 2 dialog box opens.

4. Select the file name of your new project and designate its directory path. A message appears if the directory does not exist. Click Yes to create the new folder.
   
   The Create a Project - Step 2 of 2 dialog box opens.

5. Type any directories that contain files you want to use.

6. Click the Finish button when you are done.

Overview of Developing WebFOCUS Maintain Applications
WebFOCUS Maintain creates a briefcase for your project with four components, HTML Files, Maintain Files, Master Files, and Procedures, and places it in the Explorer, as shown in the following image.

For a more complete overview of setting up your project, see Setting Up Your WebFOCUS Maintain Project in the Developing WebFOCUS Maintain Applications manual.

**Step 2: Describing the Data**

The purpose of most WebFOCUS Maintain applications is to manipulate data in a data source. Before WebFOCUS Maintain can do this, it has to know what the data in the data source looks like. It does this by using a data source description (also known as a Master File).

There are two methods for putting a data source description into your project, depending on whether it already exists:

- When you set up your WebFOCUS Server, you may have created data source descriptions for your data using the Server Console. This is the preferred method for creating data source descriptions.

  1. Open the Master Files folder.

  2. Turn on Show All Files, either by clicking the Display all files button in the Exploring toolbar, or by clicking Show All Files in the View menu.

  3. Right-click the data source description and click Add to Project.
If a data source description is in your project path, you can easily add the data source description to your project.

**Note:** You can easily add directories to your project path by right-clicking the project, clicking Properties to open the Properties dialog box, and clicking the Directories tab.

If the data source description is not in your project path, open WebFOCUS Environments (at the bottom of the left pane of the Explorer window). It contains your local WebFOCUS development server (localhost), as well as any other WebFOCUS Servers you have set yourself up to access. Open the WebFOCUS Server where the data source description resides and find it (usually in Data Servers\EDASERVE\Applications). Then drag it into your project folder.

- If you need to create data source descriptions from scratch, you can design them on your desktop using the Master File Editor. For more information on the Master File Editor, see the *Describing Data With Graphical Tools* manual.

**Step 3: Logically Partitioning the Application**

Your next step is to plan which platform should run each piece of logic in your application. This is called logically partitioning the application.

In general, you must classify each piece of logic into one of two types:

- **Front-end logic.** This generally includes forms, Web links, VBScripts, and JavaScripts.
- **Data access logic.** This is the part of your application that extracts data from the data source and writes data to the data source.

If you have more than one WebFOCUS Server in your environment, you must determine on which server each piece of data access logic needs to run.

Logically partitioning the application enables you to place each procedure on a different WebFOCUS Server easily. The Maintain language contains a special command, CALL, which enables one procedure to run another procedure at any WebFOCUS Server.

To create a procedure, follow these steps:

1. Right-click the project or the Maintain files folder in the Explorer, click New in the shortcut menu, and then click *Project File*. 

Overview of Developing WebFOCUS Maintain Applications
2. In the Create new project file dialog box, type a name for your Maintain procedure and click OK, as shown in the following image.

![Create new project file dialog box](image)

The Maintain Procedure Editor opens, displaying code for the newly created project file.

3. Close the window.

**Step 4: Creating the Front End**

Your next step in developing an application will probably be to create the front end, meaning the user interface with which the end user interacts. Your user interface is made up of forms, which you develop using the Form Editor.

To create a form:

1. Select the procedure you want the form to be in (a form is one of the components of a procedure).
2. Right-click the procedure in the Project Explorer, click New in the shortcut menu, and click Form in the submenu.

One of your procedures is designated as the starting procedure, which means that WebFOCUS Maintain runs this procedure to run your application. You probably want the opening form of your application to be in this procedure (although it does not have to be here). You use the project Properties dialog box to specify the starting procedure.
WebFOCUS Maintain supplies you with a form (named Form1) in every procedure and supplies the code to display this form immediately after running the application. Using the name Form1 in the initial form of your application is recommended.

After you have created your forms, edit them using the Form Editor, as shown in the following image.

Here is a brief overview of how to create forms using the Form Editor:

- The Controls Palette contains the controls that you can place on your form, as shown in the following image.

These controls represent various graphic elements that you are probably familiar with (text boxes, radio buttons, and check boxes).
To place a control on the form, first click the control on the Controls Palette. Then move your cursor to the form and click and drag a rectangle that defines the size of the control.

After you place controls on a form, you can easily move them around and resize them. The Form Editor also contains many alignment tools that enable you to place your controls exactly where you want them.

- The Properties sheet displays the properties of the selected control or, if no control is selected, the properties of the form itself, as shown in the following image.

![Properties Window](image)

Properties define how form and control appear, and how they behave at run time.

To change a property, select it in the list and then click the box at the right and type a new value. To change some of the properties, you must open a dialog box. Do this by double-clicking the property name.

You can also set many of these properties at run time by dragging them into an open text-editing window.
The Event Handler editor is where you define what happens when an end user performs an event in your application. An example of the Event Handler editor is shown in the following image.

An event can be anything done by the end user, such as opening a form, clicking a button, or typing text in a field. You use the Event Handler editor to assign these events to some action, such as retrieving data from a data source, opening another form, writing data to a data source, or verifying a value.

You can open the Event Handler editor in one of the following ways:

- Use the Events tab in the Property sheet.
- In the Form menu, click *Edit Event Handlers*.
- Right-click any form or control and, in the shortcut menu, click *Edit Event Handlers*.
- Click the *Edit Event Handlers* button in the Layout toolbar.

After you open the Event Handler editor and select a combination of a control and an event, you can assign one of the following actions:

- **Perform a Maintain function.** Click the *Maintain Function* button in the Event Handler editor, create a function in the Project Explorer, and then drag it into the Event Handler editor.
Perform a JavaScript. Click the JavaScript button in the Event Handler editor and type the code in the entry box.

Perform a VBScript. Click the VBScript button in the Event Handler editor and type the code in the entry box.

Access a Web link. Click the Web Link button in the Event Handler editor, and select a Web link or create a new one.

Close the form. Click the Close Form button in the Event Handler editor.

Close the application. Click the Close Project button in the Event Handler editor.

For more information on the Form Editor, see Using the Form Editor, Developing and Using Forms, Defining Events and Event Handlers, and Developing and Using Controls in the Developing WebFOCUS Maintain Applications manual.

Step 5: Creating the Data Access Logic

Your next step in developing the application is to create the data access logic, meaning the code that extracts data from the data source, manipulates it, and writes it back to the data source.

You code all of your data access logic using the Maintain language. The Maintain language is a robust yet simple, object-based 4GL. It is consistent across all platforms while incorporating the functionality of a 3GL and the data access capabilities of SQL.

You must specify which data sources you want a procedure to access before writing any code to read or write to that data source.

Procedure: How to Make a Procedure Access a Database

Specify which data sources you want a procedure to access.

1. If you want to put your data access code in a separate procedure (to take advantage of n-tier processing), create a procedure in your project.

2. In the Project Explorer, right-click the procedure, and then click Use data sources.

3. If you wish, use the Display all files button to view all the files in the project path.
4. Select the data sources this procedure will access and click OK, as shown in the following image.

![Use these Data Sources in Procedure GetAccData](image)

**Procedure: How to Write Maintain Language Code**

1. Double-click the procedure to open the Procedure Editor.
2. Between the CASE Top and END keywords, specify the code that your procedure will run. You can do this in either of the following ways:
   - Right-click in the window and click **Language Wizard** in the shortcut menu. Then follow the instructions to generate your Maintain language code.
   - Type the code that your function runs. For more information, see **Command Reference** in the Maintain Language Reference manual.

   You can obtain context-sensitive Help by selecting any keyword and pressing the F1 key. Notice also that your procedure code is color-coded.

**Step 6: Setting Up Front End and Data Access Procedures to Call Each Other**

After you have written the front-end and data access procedures, you must set them up to call each other, and pass data back and forth. Since all variables are local, meaning defined only within the context of a procedure, you must pass these variables back and forth.
For example, suppose you have an application running on a Windows web server that accesses accounting data on UNIX and inventory data on MVS. A procedure called GetAccData, accesses the data on the UNIX machine and a procedure called GetInvData, accesses the data on the MVS machine. Both of these procedures pass data back to the Start procedure on the Windows machine, which displays a form with this data.

**Procedure: How to Set Up Procedures to Call Each Other**

1. Declare all variables in the calling procedure and called procedure.
2. In the called procedure, use the Procedure Parameters dialog box to declare what parameters it expects to receive and what data it passes back to the calling procedure. To open this dialog box, right-click the procedure and in the shortcut menu, click *Add parameters*, as shown in the following image.

![Procedure Parameters Dialog Box](image)

3. In the calling procedure, use the CALL command. The parameters of the CALL command determine the data that gets passed to the called procedure and what data gets passed back. You can easily place a CALL statement in your procedure using the Language Wizard.

   For more information, see *Command Reference* in the *Maintain Language Reference* manual.

**Step 7: Adding Other Application Components**

WebFOCUS Maintain enables you to incorporate many other types of files into your application. Here are some of the most common:
JavaScripts and VBScripts are standalone functions that are embedded in your webpages for execution at the browser level.

JavaScripts and VBScripts can perform local validation. Local validation enables validating user interaction on a webpage without having to return to the web server.

You can include script libraries, and then assign functions in the script libraries to event handlers in the Event Handler editor.

For more information, see Defining Events and Event Handlers in the Developing WebFOCUS Maintain Applications manual.

Procedures are WebFOCUS procedures, written using the WebFOCUS language. They generate reports, manipulate your environment, and run other non-WebFOCUS Maintain programs, such as 3GL programs (COBOL, C, C++), CICS transactions, IMS/TM transactions, and RDBMS stored procedures (from Oracle or Sybase, for example).

HTML Files are hypertext documents that are portable from one platform to another. There are several ways you can use an HTML file in your application:

- Use an HTML file as a launch page to start your WebFOCUS Maintain application. For more information, see Running WebFOCUS Maintain Applications in the Developing WebFOCUS Maintain Applications manual.
- Use the Frame control to place an HTML file on a form.
- Use the Help property to open an HTML file in a form.

Java applets are compiled Java programs that can perform interactive animation, immediate calculations, or other simple tasks without having to send a user request back to the server. You include Java applets on a form using the Java applet control.

Images improve the appearance and usability of your forms.

**Procedure:** How to Use a WebFOCUS Procedure in Your Application

If the WebFOCUS procedure already exists, you can simply include it in your application.

1. Open WebFOCUS Environments (at the bottom of the left pane of the Explorer window). It contains your local WebFOCUS development server (localhost), as well as any other WebFOCUS Servers you have set yourself up to access.

2. Open the WebFOCUS Server where the WebFOCUS procedure resides and locate it (usually in Data Servers\EDASERVE\Applications).

3. Drag it into your project folder.

4. Return to WebFOCUS Developer Studio.
5. You will see your project listed in the WebFOCUS Developer Studio Explorer. Create a new procedure in the Procedures folder and use any of the WebFOCUS Developer Studio tools to create code.


7. In the Field Properties dialog box, click the Drill Down tab.

8. In the Drilldown Definitions drop-down menu, select Maintain Procedure.

9. Type the name of the Maintain procedure in the Procedure Name box.

10. Optionally, you can create parameters to pass to the Maintain procedure using the With Parameters section.

11. Open the Maintain procedure or the Event Handler editor.

12. Use the Language Wizard to specify that you want to create code to Run another procedure. You could also type in the syntax. See the description of EXEC in Command Reference in the Maintain Language Reference manual.

For more information, see Using WebFOCUS Procedures in Your Application in the Developing WebFOCUS Maintain Applications manual.

Step 8: Testing All the Application Procedures Locally (Optional)

After you have created all of your application procedures, it is time to test them. If you test your application locally before deploying it, you can determine whether an application works without worrying about the WebFOCUS Server environment.

Note: If you decide to test your application locally, you must have your data source residing locally. If you cannot easily obtain local versions of your data, skip this step.

To test your application, click the Deploy and Run scenario button on the Exploring toolbar.

If there are any errors, they are listed in the Run tab of the Output window. To find the erroneous code, double-click the syntax in the Run tab, and WebFOCUS Maintain automatically opens the appropriate editor at the line with a problem.
Step 9: Setting Up the Deployment Scenario

Your next step is to deploy your project files, which means moving the components of the project to the appropriate WebFOCUS servers and web servers for execution. Your deployment scenario determines where your project components should be moved to in order for your application to work. You can have as many deployment scenarios as you wish, so you probably have one deployment scenario using a test WebFOCUS Server, and another deployment scenario that deploys to the production WebFOCUS Server.

Procedure: How to Set Up Your Deployment Scenario

1. Create a deployment scenario by doing one of the following:
   - Select New Deploy Scenario in the list of deployment scenarios in the Exploring toolbar.
   - Right-click the project, click Project Development, and then click New Scenario.
   - In the File menu, click Deploy, and then click New Scenario.

2. In the New Scenario dialog box, give your scenario a name. If you wish, you can specify a starting object (the component that should be run first) and a default partition (the default WebFOCUS Server to which WebFOCUS Maintain assigns all components).

3. Open the Deploy folder. This folder appears in the project folder the first time you create a deployment scenario.

   Note: The Local Deploy deployment scenario, along with any deployment scenarios you created, appears in the list of deployment scenarios. Local Deploy deploys your application to your local WebFOCUS Server. You cannot edit Local Deploy.

4. Open your deployment scenario.

   The components of your project appear on the left. A list of the WebFOCUS Servers to which you have access appears on the right.
5. Expand the WebFOCUS Servers to which you want to assign components. Under each server appears a WEB/APPROOT/Project folder and an EDA/EDASERVE/APPPATH/Project (you may see others, as well). The WEB folders represent web servers and the EDA folders represent WebFOCUS Servers, as shown in the following image.

![Scenario Editor - Remote Deploy](image)

6. Drag each component of your project to the appropriate web server or WebFOCUS Server. In general, you deploy files that you create with WebFOCUS Developer Studio (WebFOCUS files, Maintain files, Master Files) to the WebFOCUS Server, and you deploy third-party files (HTML files, images, script files) to the web server. There are exceptions to this rule.

For more information on deployment, see *Partitioning and Deploying Project Files* in the *Creating Reporting Applications With Developer Studio* manual.

Your project is now ready to be deployed and run.
Step 10: Deploying and Testing the Application

Now that you have set up the deployment scenario, you can deploy and test your application.

1. Make the deployment scenario active by selecting its name in the list of deployment scenarios in the Exploring toolbar.

2. Do one of the following:
   - In the File menu click Deploy, and then click Deploy and run.
   - Right-click the application, in the shortcut menu click Deploy, and then click Deploy and run.
   - In the Exploring toolbar, click the Deploy and run scenario button.

For more information, see Running WebFOCUS Maintain Applications and Debugging WebFOCUS Maintain Applications in the Developing WebFOCUS Maintain Applications manual.
Welcome to the WebFOCUS Maintain basic tutorial. In this chapter, you are going to create a WebFOCUS FanClub application. This application enables you to maintain a data source of WebFOCUS Maintain fans. You will be able to add fans and view existing fans. This tutorial assumes no prior WebFOCUS Maintain experience.

In this chapter:

- Before You Begin
- Creating a Project
- Familiarizing Yourself With the Explorer
- Adding a Data Source to Your Project
- Familiarizing Yourself With the Project Explorer
- Designing a Form
- Saving Your Work
- Selecting a Starting Object
- Running Your Project Locally
- Using Radio Buttons
- Giving Your Form a New Title
- Writing Data to the Data Source
- Adding a Form to Display Data From a Data Source
- Using the HTML Overflow Property
- Adding Form Navigation Buttons
- Deploying Your Project to a Different WebFOCUS Server
- Adding Images to Your Project
- What Is Next?
Before You Begin

Before you begin working on the WebFOCUS FanClub application, make sure you have done the following:

- Installed and verified a WebFOCUS Server.
- Installed WebFOCUS Developer Studio on your machine (this includes the Maintain Development Environment).
- Verified the connection of your machine to the WebFOCUS Server.
- Checked to see that you have the sample tutorial files required for the WebFOCUS Maintain tutorial on your machine. This includes the following files:
  - The fannames data source (both the .MAS and .FOC files).
  - The addafan.gif, currfan.gif, fan.gif, and spiralbg.gif image files.

All of the FanClub files are installed under ibi/apps/maintain. The images are located under ibi/apps/maintain/images.

Note: When you run your Maintain application, it will open in the default Developer Studio Viewer. For more information, see the Developer Studio Application Development Getting Started manual.

Creating a Project

Before you can do any development work in WebFOCUS Maintain, you must create a project in WebFOCUS Developer Studio. The WebFOCUS Developer Studio project contains your working files. This is where you develop, test, and edit your code. When you place your project files on a WebFOCUS Server (called deploying) your project becomes an application.

You can create the project with the Create a Project Wizard. It guides you through the project creation process by prompting you for all necessary information.
Procedure: How to Create the Project Folder

1. From the Start menu, select Programs.

2. From the WebFOCUS 81 Developer Studio folder, select WebFOCUS Developer Studio to start WebFOCUS Developer Studio.

3. Do one of the following:
   - Right-click Projects on localhost and click New Project.
   - In the File menu, click New Project.

The Create a Project - Step 1 of 2 dialog box opens, as shown in the following image.

4. In the first input box, type FanClub as the name of your new project. The wizard adds the extension .GFA and appends the file name to the default directory path that appears in the second input box.

5. Click Yes when prompted to create a new directory.

The Create a Project - Step 2 of 2 dialog box opens.
6. Click the *Add* button and browse to locate the Maintain directory, which is where the sample (tutorial) files are located. Select it so that it is highlighted, and click *OK*, as shown in the following image.

![Create a Project - Step 2 of 2](image)

7. Click the *Finish* button. The new folder is added to the Explorer.

**Procedure: How to Create the Maintain Procedure**

1. Right-click *FanClub*, click *New*, and then click *Maintain Procedure*.

   or

   In the FanClub folder, right-click the *Maintain Files* folder, click *New*, and then click *Maintain Procedure*. 
The Add Maintain Procedure dialog box opens, as shown in the following image.

2. Type the name of the procedure and click Open.
The procedure opens in the Maintain Development Environment, as shown in the following image.

At the top of the window, underneath the menu bar, you will see toolbars with buttons for performing commonly used commands. (Later, you will have the option to customize these toolbars.)

Below the toolbars you should see the text editor, displaying the default code for Start, the Explorer window containing the contents of the FanClub project, and the Project Explorer, displaying the Maintain procedures in your project.

**Familiarizing Yourself With the Explorer**

The Maintain Development Environment displays two views of FanClub; in the Explorer window and in the Project Explorer window. For now, close the text editor containing the code for Start.mnt and concentrate on the Explorer.
If you already know how to use WebFOCUS Developer Studio, the Explorer should be familiar to you. The Explorer is shown in the following image.

The left pane of the Explorer displays the contents of Developer Studio. The first item, Projects on localhost, displays the projects you are developing. FanClub, the project you created, is listed first, followed by SESSION, the default WebFOCUS project.

Under Projects on localhost, is a list of WebFOCUS Environments, which displays the WebFOCUS Servers available to you. Depending on how you have configured your computer, one or more WebFOCUS Servers are listed. You will learn more about these servers later when you deploy your project.

The right pane of the Explorer displays the contents of whatever you select in the left pane. It should be displaying the contents of the FanClub project. The default folders for a project are HTML Forms, Maintain Files, Master Files, Procedures, and Other, but you can add new ones.

Frequently, you use previously existing files in a project, and the Explorer enables you to easily view all files available in the project path. You can also easily turn off this view to view only the files that have been included in the project.

**Procedure: How to View the Files Available to a Project**

1. Double-click any of the folders in either Explorer pane except for Maintain Files (either HTML Forms, Master Files, or Procedures).
2. In the toolbar, click on Display all files in the project path button.

If this button is not available, click Show All Files in the View menu. The default is for Display all files to be on, but your setting may be different.

As you turn Display all files on and off, you should see files appearing and disappearing in the right pane of the Explorer. These files are available for you to use in constructing your project, but none of them are actually included in the project yet.

Adding a Data Source to Your Project

The first step you take when designing a project is to define the data sources on which your final application is going to operate.

WebFOCUS Maintain applications can operate on data from many types of data sources, using the capabilities of the WebFOCUS Server. To do so, an application needs the following:

- An Information Builders data source description, which describes the structure of the data (what the fields are, what their format is, and so on) and where to find the data.
- Data sources that contain the actual data. WebFOCUS Maintain applications cannot use these data sources without the data source description telling them how the data is organized.

To define which data sources your application is going to operate on, you include its data source description in your project. There are two ways to get a data source description into your project:

- Create a new data source description.
- Use an existing data source description.

Tip: The WebFOCUS Server ships with utilities that create data source descriptions (even from other database management systems, such as Oracle). In general, the only reason you need to create data source descriptions is if you are creating a brand new data source.

For this tutorial, you will use existing data source descriptions.

Procedure: How to Add a Data Source to Your Project

1. If you have not already done so, open the Master Files folder in the Explorer and turn on Display all files.
2. Find the fannames.mas data source (you may need to adjust the size of the window so that you can view the scroll bars).
3. Right-click fannames.mas and click Add to Project.
Note that the icon to the left of fannames.mas is now displayed in color, while the rest are in black and white.

If you turn off Display all files, only fannames.mas is visible.

**Familiarizing Yourself With the Project Explorer**

The Project Explorer, which by default appears on the left side of your screen, displays another view of your project.

Currently, the only component here is Start, but as you continue working on the FanClub project, more components appear here.

If you expand Start, you see the following:

There is a folder for all of the forms in Start. Currently there is one form in it, Form1. If you expand it (not open it), you will see the text *There are no items to show*. When you create Start, this form is created by default.

The next component is called Top. Top is a *function*, which is a series of commands in a procedure grouped together as a unit of control. A function accomplishes some small task, such as calculating values, extracting data from a data source to place in a data source stack, or writing information to a data source.

Every procedure has a Top function, which consists of the command that runs immediately when the procedure runs. WebFOCUS Maintain automatically creates the Top function when you create a Maintain procedure.

**Procedure:**  **How to Use a Data Source in a Procedure**

1. In the Project Explorer, right-click Start, and then click Use data sources.

2. In the Use these Data Sources in Procedure Start dialog box, select fannames in the list of available data sources.
3. Click the right arrow button \( \gg \) to move fannames into the list of Data sources to use, as shown in the following image.

![Image of Use these Data Sources in Procedure Start dialog box]

4. Click OK.

**Viewing the Structure of a Data Source in the Project Explorer**

After you use the fannames data source in the Start procedure, its name appears in the Project Explorer underneath the Data Sources folder with a plus sign (+) next to it.

If you click the plus sign (+) next to fannames, WebFOCUS Maintain displays the segments in the fannames data source:

![Image of Data Sources with plus sign]

A segment is a collection of fields that have a one-to-one relationship to each other. The fannames data source has one segment, called CUSTOMER.

**Note:** This data source has only one segment. Hierarchical data sources, such as FOCUS, can have more than one segment. If you would like to see data sources with more than one segment, migrate the CAR data source description (one of Information Builders standard sample files) into your project and view it. You can then delete the CAR data source description from your project by selecting it and clicking *Delete.*
If you click the plus sign (+) next to CUSTOMER, you see the fields in that segment, as shown in the image below.

![Diagram showing fields in the fannames data source]

Since fannames has only one segment, these are the only fields in the fannames data source. The key next to the SSN field indicates that SSN is a key field, which means it uniquely identifies the segment instance. In the fannames data source, each fan has one SSN, and no other fan should have the same number.
Designing a Form

An example of the first form that you will create for the FanClub application is shown in the following image.

This form enables end users to add a new fan to the data source.

The information typed into this form will be written to the fannames data source, so it should correspond to the fields in the data source. The fastest way to create the fields on the form is to use the existing data source fields. Therefore, your first task is to add data source fields to the form.

**Procedure: How to Add Data Source Fields to a Form**

1. Open the Form1 form.
2. In the Project Explorer window, click the plus sign to the left of fannames to view the CUSTOMER segment.
3. Drag the CUSTOMER segment into the form, as shown in the following image.

WebFOCUS Maintain opens the Select Segment Fields dialog box. You must specify which data source stack fields you want to display on your form. First, you need to understand the role of data source stacks in your project.

**What Are Data Source Stacks?**

WebFOCUS Maintain procedures do not directly display or manipulate information in a data source. Instead, use data source stacks as intermediaries between end users and the data source.

A stack is a non-persistent (or in-memory) table where you can store and manipulate data from one or more data sources. WebFOCUS Maintain procedures use stacks to hold values you read from the data source and to manipulate data before writing it back to the data source.

Since this is a new procedure, there are no stacks yet. You are going to create one named AddFanStack, instead of using the default CustomerStk. The structure of the stack is based on the fields in the data source, in other words, the stack is going to have the columns SSN, LASTNAME, FIRSTNAME, and so on. However, this stack will be empty until you do something to put data in it.

**Procedure:** How to Specify a Data Source Stack for Fields on a Form

1. In the Select Segment Fields dialog box, select New Stack. This specifies where your procedure stores the data to be displayed in the fields. An input field appears under the New Stack radio button.
2. In the New Stack field, type *AddFanStack*.

3. Scroll down to the bottom of the list of fields and deselect *TITLE* and *USER*, as shown in the following image.

![Select Segment Fields](image)

4. Click *OK*.

WebFOCUS Maintain places all of the fields from the CUSTOMER segment except TITLE and USER into the form.

**Components of a Procedure in the Project Explorer**

In the Project Explorer, there have been some additions to the components of Start:

- If you expand Form1, you see a list of fields that you placed on the form.
The stack AddFanStack has also been added to Start. If you expand AddFanStack, you will see the columns in this stack, as shown in the following image.

Working in the Form Editor

In the Form Editor, you design the user data adapters for your applications. The Form Editor design tools are identified in the following image.

Note: The screen may look different from what you see here. The windows have been moved.

The following are the main components of the Form Editor:

- The Form is your main working area. What you place here appears on the form your end users see.
The Maintain Controls palette contains the controls that you can place on your form, such as text boxes, buttons, radio buttons, check boxes, and so on. Controls are objects on the screen that you can maneuver to execute an action. You can use many of these controls in this tutorial.

**Note:** When you select an object, the mouse pointer changes to a crosshair so that you can draw it on a form. You can use the arrow keys to move the object to a specific location.

The property sheet is where you specify information about controls that you have placed on the palette. You will always see the properties for the currently selected control here. If no control is selected, you see the properties for the form itself.

The Layout toolbar contains alignment and grouping commands.

**Note:** The layout toolbar usually appears at the bottom of the application. If it is not visible, click the Tools menu, click Customize toolbars, then select Layout.

### Moving Controls on a Form

When you placed these fields on the form, WebFOCUS Maintain placed them in a column. This is not an optimal arrangement for the fields. Rearrange them so that the Firstname and Lastname fields are next to each other in a row, and City, State, and Zip fields are also next to each other.

**Procedure: How to Move Controls on a Form**

1. If the grid for your form is not set to **on** (the default setting), then turn it on either by selecting Grid from the View menu or by clicking the Toggle grid button on the Layout toolbar. Make sure the form is the active window.
   
   If the grid is on, when you move controls on your form, the Form Editor automatically snaps them to the closest grid point. This feature enables you to keep controls easily aligned on a form.

2. Move Lastname (by clicking on it and dragging it to the new location) far enough to the right so that you can move Firstname up and to the left of it.

3. Move the Firstname, Company, Address, and City fields up the grid.
   
   Since the grid is on, you preserve the correct vertical spacing between fields.

4. Drag State to the right of City.

5. Drag Zip to the right of State.
6. Move the *Phone*, *Email*, and *Enrollment date* fields up.

7. Right-click the form. From the pop-up menu, select *Tab Order* and reorder the fields to reflect the order in which they appear on your form.

An example of this is shown in the following image.

---

**Renaming Field Prompts**

While *Firstname* and *Lastname* are certainly descriptive field prompts, they are definitely not English words. Rename these two field prompts to *First* and *Last*. 

---
**Procedure: How to Rename Field Prompts**

1. Select the Firstname field.

   Notice that the control in the property sheet says (Group), indicating that the Firstname field is not a single control, but is actually two controls grouped together, as shown in the following image. It is useful to group controls so that WebFOCUS Maintain treats them as a unit. This means that you cannot move one without moving the other.

   Firstname is made up of two controls: a text box named Firstname_Text that serves as the prompt, and an edit box control named Firstname_Edit where the end user types a value for the Firstname field. Together they are called a prompted edit box.

   ![Properties](image)

2. Select **Firstname_Text** from the drop-down list at the top of the property sheet, as shown in the following image.
You can then see the properties for Firstname_Text.
3. Scroll to find the Text property (the value is Firstname), and change the value to First, as shown in the following image.

![Properties panel showing Text property changed to First](image)

4. Press Enter to confirm your change.

**Procedure: How to Quickly Rename Field Prompts**

There is a faster way to rename field prompts. Now you will rename Lastname using this method.

1. Select the Lastname prompted edit box.
2. Place your cursor over the Lastname prompt. It turns into a text editing I-beam.
3. Change the prompt to Last.

**Aligning Controls**

If you have been moving these controls around in the window and renaming prompts, your fields have probably become misaligned. The Form Editor enables you to align controls easily.

One of the alignment tools is designed especially for aligning entry fields such as these, which are made up of a text control and an edit box control.
Procedure: How to Align Entry Fields

1. Select all of the entry fields on the left side (all of the entry fields except for Last, State, and Zip).

You can multiselect fields by selecting the first one and then holding down the Shift key or the Control key while you select the remaining fields.

Notice that one of the fields has green handles, while the other ones have blue handles. The field with green handles is the anchor control, meaning the field with which all the other fields are aligned, as shown in the following image.

2. In the Layout menu, click Align, and then click Center vertically in the submenu.

or

In the Layout toolbar, click the Align Edges button.

Then click the Center vertically button in the pop-up toolbar.

Note: You can adjust the fields manually to refine your layout.
Saving Your Work

Before you go any further, save your work.

Whenever you make changes to a component, WebFOCUS Maintain places an asterisk (*) next to the component name in the Project Explorer and in the title bar.

**Procedure: How to Save Your Project**

Do one of the following:

- In the File menu, click Save.
- Press Ctrl+S.
- In the General toolbar, click the Save button.
Selecting a Starting Object

The Set Starting Object dialog box is shown in the following image.

One of your procedures is designated as the starting procedure. When you execute the project, WebFOCUS Maintain executes the starting procedure first.

1. Open the Properties dialog box for the project.
2. Select the Deployment tab.
3. Click the Set button to display the Set Starting Object dialog box.
4. Select the Starting Object and click OK.
5. Click OK again to exit the Properties dialog box.

Note: For more information on selecting a starting object, see How to Select a Starting Object on page 56 and How to Run Your Project Locally on page 56.
Running Your Project Locally

Run your project to see how it looks.

**Note:** Whenever you deploy or run your project, WebFOCUS Maintain saves it.

Selecting a Starting Object

A deployment scenario needs a starting object in order to run. In this example, one of your procedures is designated as the starting procedure. When you execute the project, WebFOCUS Maintain executes the starting procedure first.

**Procedure: How to Select a Starting Object**

1. Right-click FanClub and from the pop-up menu, select Properties.
2. In the Properties dialog box, select the Deployment tab.
3. From the Starting Object drop-down menu, select the procedure you wish to use as your starting procedure (in this example, select Start).
4. Click OK.

You are ready to run and deploy your project locally.

**Procedure: How to Run Your Project Locally**

1. To run your project, do one of the following:
   - In the File menu, click Project Deployment, then click Deploy and Run.
   - Right-click the project, click Project Deployment, then click Deploy and Run.
   - On the Exploring toolbar, click the Deploy and Run button.
WebFOCUS Maintain performs a local deploy, as shown in the following image.

2. When WebFOCUS Maintain finishes deploying your project, the Cancel button at the bottom of the Deploying Application dialog box turns into a Close button. Click Close.
WebFOCUS Maintain opens the application, as shown in the following image.

This application enables you to type information into the fields displayed in the window (which places data into the data source stack AddFanStack). Next, you will improve the functionality and appearance of an application.

For now, click the close box located on the upper-right corner of the application to end it before continuing.

**Using Radio Buttons**

When you drag the fields from the data source into the form, deselect the TITLE and USER fields so they do not appear in the form.

By not copying TITLE into the form, you prevent users from typing an arbitrary title in the field. Give them the option to type either Mr., Mrs., or Ms., using a radio button control. This prevents them from typing arbitrary titles.

The task of using radio buttons can be separated into three steps:

1. Adding the group of radio buttons to your form.
2. Adding tool tip text to the group of radio buttons. This step is optional, but it demonstrates how you can display useful information to your end users.
3. Binding the results of the selection to a stack.

**Procedure: How to Add a Group of Radio Buttons to Your Form**

1. In the Maintain Controls palette, click the *Radio button* control.
2. Draw a small rectangle on the form next to Ssn, First, and Last fields.
   When you let go of the mouse button, you see the green handles indicating that the control is selected. If you drew the rectangle large enough, you might see one radio button and the description of this radio button, as shown in the following image. (If you see nothing, just enlarge the size of the control.)

3. Double-click the control to open the List Source dialog box, as shown in the following image.
4. Double-click the highlighted `RadioButton1` text to open the Enter a List Item dialog box, as shown in the following image.

5. Replace the text `RadioButton1` with the text `Mr`. You do not need to type anything in the Assigned value field.

6. Click `OK` or press `Enter`.

7. Insert a new value by clicking the plus icon (+) in the List Source dialog box.

8. Type `Mrs.` and click `OK` or press Enter.

9. Repeat steps 7 and 8 to type the value `Ms`, as shown in the following image.
10. Click OK.

Your radio buttons should look like this (you may need to adjust the size of your control to see all three of them):

![Image of radio buttons]

**Procedure: How to Add Tool Tip Text**

1. Make sure you have selected your group of radio buttons.
2. In the Properties sheet, click the Properties tab.
3. Select the ToolTipText property.
4. In the empty field to the right, type *Please select one* and press Enter, as shown in the following image.

![Properties sheet with ToolTipText property highlighted]

You can move your cursor over the radio buttons in the Form Editor to see the tooltip.

**Procedure: How to Bind the Results of the Selection to a Stack**

1. Make sure you have selected your group of radio buttons.
2. In the Properties sheet, double-click the *SelectedItem* property to open the Binding the Selection Result dialog box. The binding box dialog box is shown in the following image.

![Binding the Selection Result dialog box](image)

3. Expand *AddFanStack* by clicking the plus sign (+) to its left.
4. Select the *TITLE* field.
5. Make sure *Text* is selected for Send the result as.
6. Click *OK*.

Now, when end users select either Mr., Mrs., or Ms. in this form, their choices are saved in the TITLE column of AddFanStack.

If you wish, save your work and deploy and run your application again to see how it looks. Make sure to close the application before continuing the tutorial.

**Stacks and Implied Columns**

When you created *AddFanStack* using the Select Segment Fields dialog box (see *How to Add Data Source Fields to a Form* on page 44), you deselected the TITLE and USER fields. Since you deselected these fields, why are they showing up as columns in the data source stack?
These fields are showing up as columns in the data source stack because of the way WebFOCUS Maintain defines data source stacks with the INFER command. When you use a field in the data source to define a column in a data source stack, WebFOCUS Maintain defines columns based on the rest of the fields in that data source. If your data source is hierarchical, the rest of the fields in the segment and the key fields in any parent segment.

WebFOCUS Maintain includes all of these fields in your data source stack so that when you update your data source from the data source stack, it knows the path to these fields.

These columns are called implied columns.

There is one field from the fannames data source that you have not placed on the form: the USER field. The end user will not be entering this field. Instead, it will be generated by the application.

**Giving Your Form a New Title**

When you ran your application, you probably noticed that the title bar said Untitled. You can change this so that your application displays a more useful title at run time.

**Procedure: How to Give Your Form a Title**

1. Deselect all controls in your form (that is, click anywhere in the form that is not a control).
2. In the property sheet, scroll down to the Title property and select it.
3. Change Untitled to Add a New Member, as shown in the following image.
If you wish, save your work and deploy and run your application to see how it looks. Make sure to close the application afterwards before continuing the tutorial.

**Writing Data to the Data Source**

Your next step is to enable end users to write the data that they type in this form to the data source. They do this by clicking a button that says *Add* at the bottom of the form. You add the button to your form and write the Maintain Language source code that inserts the data into the data source.

**Procedure: How to Add a Button to Your Form**

1. In the Form Editor, select the *Button* control on the Control palette.
2. Move the cursor to the bottom of your form and draw a rectangle where you want your button to go.
3. Change the text in the button to *Add*. The text in the button is automatically selected when you create the button, so all you have to do is type your new text, as shown in the following image.

![Add Button](image)

4. In the property sheet, notice that the first property is *(Name)* and its value is Button1. This is the internal program name of the button (how WebFOCUS Maintain refers to it). By default, WebFOCUS Maintain names forms and controls with their type and a unique number. If you are planning to refer to a form or control in other places in the procedure, giving it a more descriptive name is recommended.

5. Change the name of the button to *AddButton*.

**Writing Functions**

Now that you have created a button that the end user will click, you must create the code to run when the end user clicks the button.

You put this code in a function. A function is a series of commands in a procedure grouped together as a unit of control. A function accomplishes some small task, such as calculating values, extracting data from a data source to place in a data source stack, or writing information to a data source.

**Procedure: How to Write a Function**

1. In the Project Explorer, select the *Start* procedure.
2. Right-click Start. In the shortcut menu, click New. Then click Function (Case) in the submenu.

3. In the New Function dialog box, give your function the name *AddFan*, as shown in the following image.

![New Function dialog box](image)

4. Click OK.

Your new function appears in the list of components of the Start procedure, as shown in the following image.

![Start procedure components](image)

5. Double-click *AddFan*.

WebFOCUS Maintain opens the source code for your function in the Procedure Editor.
About the Procedure Editor

Underlying many of the graphical elements of your WebFOCUS Maintain project is Maintain language source code.

You can edit this source code directly using the Procedure Editor.

Take a brief look at the source code for the Start procedure, compared with the components that you have added so far in the application development process.

Note: The text in your window may wrap differently.

The Start procedure starts with the line

MAINTAIN FILE fannames

and ends with the keyword

END

All WebFOCUS Maintain procedures start with the keyword MAINTAIN (which must be typed in uppercase) and end with the keyword END (also uppercase). When you create a procedure, WebFOCUS Maintain puts these two keywords in for you automatically.

The second and third words, FILE fannames, tell WebFOCUS Maintain what data sources this procedure is going to access. If you look in the list of project components, you see a Data Sources folder and, under this, the fannames data source.
Following this line is a comment line beginning with $$$. If you have syntax coloring turned on, you can tell that this is a comment (green is the default color for comments, although you may have a different setting). Turn syntax coloring on and off using the Editor tab in the Options - Maintain Development Environment dialog box (in the Tools menu, click Environment options). This comment line is automatically generated when you create a procedure.

This particular comment, $$Declerations, is generated automatically by WebFOCUS Maintain when you created the procedure. If you create any variables, WebFOCUS Maintain places their source code after this comment.

The next line, Case Top, begins the definition of the Top function. This definition takes up several lines and ends with the keyword EndCase. The first statement, which begins with Infer, defines the AddFanStack data source stack.

The next line of the Top function

Winform Show Form1;

is the code that displays Form1 at run time. This code was generated automatically when you created the procedure.

The next three lines begin with the characters -* and are also comments. They contain an alternate piece of code for opening a form:

Winform Show_And_Exit Form1;

This code opens Form1 and then exits the application while still displaying Form1.

The final two lines before the END keyword that ends the procedure

Case AddFan
EndCase

are what define the function you just created. You are going to add some code to this function using the Language Wizard.

**Procedure:** How to Build Maintain Language Code Using the Language Wizard

1. Ensure that your cursor is in the line between Case AddFan and EndCase.
2. Right-click in the Procedure Editor window.
3. From the shortcut menu, select Language Wizard.

   The Maintain Language Wizard opens. The Maintain Language Wizard helps you build Maintain language source code without typing the syntax yourself.

   The first Language Wizard window asks you to specify, in general, what kind of task you want to accomplish.
4. Select *Update records in a data source*, as shown in the following image.

![Maintain Language Wizard](image)

5. Click *Next*.

Now that you have specified the general task you want to perform, the Language Wizard narrows down the task further. Notice that after each task, there is a word in parentheses. This is the name of the Maintain Language command that executes that task. Notice also the box at the bottom of the window contains the Maintain language code being generated by the Language Wizard. As you move through the Language Wizard, you see more code.
6. Select *Add one or more new data source records (Include)*, as shown in the following image.

![Maintain Language Wizard dialog box](image)

7. Click *Next*.

Now that you have specified which command to use, the Language Wizard asks you to supply the parameters for that command. In this case, you must tell it which data source is being updated and from where.

You first specify which data source is being updated.

**Note:** The Available fields list contains the data sources that you are using in this procedure, not the list of data sources in the project.

8. Expand the fannames data source.

9. Expand the CUSTOMER segment.
10. Copy the SSN field into the Fields to update box by clicking SSN and then clicking the right arrow button or by double-clicking SSN. Notice that, as with stacks, all the other fields in the CUSTOMER segment are also copied, as shown in the following image. This is because the Maintain language assumes that if you are adding new data source records to a data source, you want to write information into all the fields in a segment. For more information, see *Stacks and Implied Columns* on page 62.

Notice that the Maintain language box at the bottom now reads:

```
Include fannames.CUSTOMER.SSN;
```

11. Click Next.

Your final step is to indicate where this data is being written from, which in this case, is AddFanStack.

12. Select Stack.

13. Choose *AddFanStack* from the list.
14. Leave the 1 in the Starting from row field and select the *All the records in the selected stack option*, as shown in the following image.

Notice that the Maintain language box at the bottom now reads:

```
For all include fannames.CUSTOMER.SSN from AddFanStack;
```

15. Click *Finish*.

WebFOCUS Maintain places the source code that the Language Wizard generated in between the Case AddFan and EndCase lines.

**Clearing Data From Stacks**

Now that you have included the data from AddFanStack into the fannames data source, it is a good idea to clear the data from AddFanStack. Use the Language Wizard to write this code.
**Procedure:**  How to Clear the Data From a Stack Using the Language Wizard

1. Place the insertion point after
   
   ```
   For all include fannames.CUSTOMER.SSN from AddFanStack;
   ```
   
   but before
   
   ```
   EndCase
   ```

2. Right-click in the Procedure Editor window and, in the shortcut menu, click *Language Wizard*.

3. Select *Operate on a stack* and click *Next*.

4. When the Language Wizard asks you which stack operation you would like to perform, select *Clear the contents of a stack* and click *Next*.

5. The Language Wizard asks you to select one or more stacks to clear. Select *AddFanStack* and click *Finish*.

   ```
   Stack clear AddFanStack;
   ```

**Assigning the Function to the Add Button**

Now that you have written the code that inserts end user data into the data source, you need to designate that when the end user clicks the Add button, this function is performed. You do this using the Event Handler editor.

An event is something that an end user performs, such as opening a form, clicking a button, or moving to a field. When you open the Event Handler editor, you see a list of controls on the form and another list of events that can happen to that control, such as clicking, double-clicking, and so on.

Select a combination of a control and whatever happens to it. Then, in the box below, type the code for what should happen when this event occurs.

**Procedure:**  How to Assign a Function to an Event

1. Double-click the *Form1* form in the Project Explorer to display the form.

2. In the Form Editor, double-click the *Add* button to open the Event Handler editor.

   Notice that AddButton is selected in the list of controls.

3. In the list of events, select *Click*. Notice the code that appears in the Event Handler editor:

   ```
   Case OnAddButtonClick
   EndCase
   ```

4. Make sure you can see both the Event Handler editor and the Project Explorer.
5. Drag the `AddFan` function from the Project Explorer into the Event Handler editor between

```java
Case OnAddButton_Click

and

EndCase
```

The Event Handler editor looks like the following:

![Event Handler Editor](image)

6. Close the Event Handler editor.

7. When WebFOCUS Maintain prompts you to save your procedure, click Yes.

8. Deploy and run your application to see how it looks.
9. Add your name to the fannames data source. You are now included in the WebFOCUS FanClub application, as shown in the following image.

![Image of a form with text fields for name, address, and phone number.]

10. Close the application before continuing the tutorial.

Adding a Form to Display Data From a Data Source

The FanClub application can add names to the fannames data source, but end users do not get much visual feedback from this task. The FanClub application needs to display the contents of the fannames data source.

This task is divided into the following steps:

1. Add a new form to the project.
2. Build the code to extract the data from the fannames data source using the Language Wizard.
3. Design the form to display the contents of fannames.
4. Create a link from Form1 to the new form.

**Procedure: How to Add a New Form to Your Project**

1. Right-click the *Forms* folder in the Start procedure, and in the shortcut menu, click *New form*. 
WebFOCUS Maintain opens the form in the Form Editor.

2. Change the name of the form from Form2 to ShowFan by right-clicking Form2 in the Project Explorer and clicking Rename.

3. Change the title of the form from Untitled to Show Fan Club Members.
   (Change the value for the Title property in the property sheet.)

**Extracting Data From a Data Source Into a Stack**

Your next step is to create a function named GetFans, which extracts all the information in the fannames data source and places it into a stack named GetFanStack. Try this yourself or see the next section for instructions.

**Tip:** Create the function and then use the Language Wizard to generate the code.

**Procedure:** How to Extract Data From the Fannames Data Source Into a Stack

1. In the Project Explorer, right-click the Start procedure, click New, and then click Function (Case).
2. In the New Function dialog box, name your function GetFans and click OK.
3. In the Project Explorer, double-click your new function to open it in the Procedure Editor.
4. Make sure your insertion point is placed after the statement
   ```maintain
   Case GetFans
   ```
   but before
   ```maintain
   EndCase
   ```
5. Open the Language Wizard (right-click in the Procedure Editor window and, from the shortcut menu, click Language Wizard).
6. Select Retrieve records from a data source and click Next.
7. When WebFOCUS Maintain asks you how you would like to retrieve records from the data source, select Starting from the current record position (Next) and click Next.
   The Maintain language contains two commands to retrieve data from a data source. This window determines which one you want.
8. When WebFOCUS Maintain asks you to select the data source segments or fields whose records you want to retrieve, in the Available fields box, expand the fannames data source, expand the CUSTOMER segment, and move the SSN field to the Fields to retrieve box. Then, click Next.
This window is where the Language Wizard determines from which data source you are reading the data. Remember that if you select a field in one segment, the rest of the fields in that segment are also extracted. See Stacks and Implied Columns on page 62 for more information.

9. When WebFOCUS Maintain asks you how many records you want to retrieve, select All the records in the selected segment. Also, make sure Change the current data source position to the top is selected, ensuring that WebFOCUS Maintain starts from the beginning of the data source when retrieving records. Then, click Next.

10. When WebFOCUS Maintain asks you to specify a stack, type GetFanStack in the text box to create one, and make sure that Clear the stack first is selected. You can leave the default value 1 in the Place the records into the stack field. Then, click Next.

11. When WebFOCUS Maintain asks you to supply any conditions, you can leave this window blank because you want to retrieve all records from the data source. Simply click Finish.

The Language Wizard should generate the following code:

Reposition fannames.CUSTOMER.SSN;
Stack clear GetFanStack;
For all next fannames.CUSTOMER.SSN into GetFanStack;

Adding an HTML Table to Your Form

You are going to display the fans from the fannames data source using an HTML table. An HTML table displays the contents of a data source stack in a read-only grid.

Another option for displaying the fans from the fannames data source is to create a report procedure in WebFOCUS Developer Studio and execute this code whenever you open this form.

Procedure: How to Add an HTML Table to Your Form

1. Double-click on the form ShowFan in the Project Explorer to display the form.

2. On the Controls palette, click the HTML Table button.

3. Draw a rectangle on the form representing roughly where you want your HTML table to go on your new ShowFan form.

   WebFOCUS Maintain opens the Control Columns dialog box, where you define the contents of your HTML table.

4. Select GetFanStack from the Stack list.

   Note: Make sure you did not select AddFanStack.

The Control Columns dialog box now displays the columns in the stack GetFanStack.
5. Copy the fields LASTNAME, FIRSTNAME, COMPANY, EMAIL, and TITLE into the Table Columns list.

Use the Move up button and the Move down button to rearrange these fields so that they are in the following order: TITLE, FIRSTNAME, LASTNAME, COMPANY, and EMAIL, as shown in the following image.

You can change the appearance of any of these columns by opening the Table Column dialog box. For example, suppose you want to change the header titles for the FIRSTNAME and LASTNAME fields so that they read First and Last.

6. Double-click Firstname.
7. In the Table Column dialog box, change the Header title to *First*, as shown in the following image.

![Table Column Dialog Box](image)

You can also change the width, justification, font, and color.

8. Click **OK**.

9. Repeat the process to change Lastname to *Last*.

10. Click **OK** to leave the HTML Table Columns dialog box.
Your form resembles the following:

![Form Example]

Creating a Link From One Form to Another

Your final step in creating your new form ShowFan is providing a way to open it from Form1. Do this by adding a button to Form1 that runs the GetFans function and opens the ShowFan form.

**Procedure: How to Link From One Form to Another**

1. Make Form1 the active window (you can select it from the Window menu or double-click it in the Project Explorer).
2. On the Control palette, click the Button control.
3. Draw a rectangle to the right of the Add button on Form1.
4. Change the text in this button to Show Fans.
5. Change the name of this button to ShowFanButton.
6. Double-click the button to open the Event Handler editor.
7. Select Click from the list of events. (ShowFanButton should already be selected in the list of form components.)
WebFOCUS Maintain adds the following code:

```csharp
Case OnShowFanButton_Click
EndCase
```

8. Drag the `GetFans` function from the Project Explorer into the Event Handler editor between these two lines of code.

9. Drag the `ShowFan` form from the Project Explorer into the Event Handler editor after `GetFans`.
   
   You see the following pop-up window.

   ![Show a Form](image)

10. Select **Show a Form**.

    Comment text appears with the `Winform Show` command that explains how to show a non-persistent form. Because you want a persistent connection for this form, leave the comment text as is, as shown in the following image.

   ![Persistent Form](image)

11. Close the Event Handler editor.

12. Click **Yes** to save your procedure.

13. Deploy and run your application to see how it looks.
14. Close the application before continuing the tutorial.

**Using the HTML Overflow Property**

When you ran your application, you may have noticed that the HTML table took up enough room to display all of the rows in GetFanStack. If you wish, you can change the HTML table so that it takes up a defined amount of space and has scroll bars so that end users can view rows that do not fit in the space.

This enables you to place information under the HTML table without having to worry about whether the HTML table overlaps it.

**Procedure: How to Add Scroll Bars to an HTML Table**

1. In ShowFan, select the HTML table.
2. In the property sheet, find the Overflow property.
3. Click the value box for the Overflow property.
4. In the list, select 2 - Scroll.
5. Deploy and run your application to see how it looks.

Close the application when you are done before continuing the tutorial.

**Adding Form Navigation Buttons**

When you ran your application and displayed the Show All Members form, you may have noticed that the only thing you can do in this window is close the form with the close box (in the upper right corner of the window) to go back to the Add a New Member form.

Add some navigation buttons to the FanClub application:

- Add a How to Add a Back Button on page 81 to the Show Fan Club Members form so that you can go back to the Add a New Member form.
- Add an How to Add an Exit Button on page 82 to the Add a New Member form.

**Procedure: How to Add a Back Button**

1. Open the ShowFan form.
2. On the Maintain Controls palette, select the **Button** control .
3. Draw a button under the HTML table.
4. Change the text on the button to *Back*. 

[81]
Adding Form Navigation Buttons

5. Change the name of the button to BackButton.
6. Double-click the button to open the Event Handler editor.
7. In the Event Handler editor, select the Click event.
8. Click the Close form button \[\text{\includegraphics[width=1cm]{close_form_button.png}}\] on the right side of the window.
   This inserts the following code in the box:
   
   ```javascript
   self.WinClose();
   ```
10. Click Yes to save your procedure.

**Procedure: How to Add an Exit Button**

1. Open Form1.
2. Place an Exit button named ExitButton at the bottom of your form.

   **Tip:** Use the Button control to draw a button at the bottom of your form. Change the text to Exit and the name to ExitButton.

3. Open the Event Handler editor.
4. In the Event Handler editor, select the Click event.
5. Click the Close project button \[\text{\includegraphics[width=1cm]{close_project_button.png}}\] on the right side of the window.
   This puts the following code in the box:
   
   ```javascript
   self.WinExit();
   ```
6. Close the Event Handler editor.
7. Click Yes to save your procedure.
8. Deploy and run your application to see how it looks.
9. Click the Exit button to close the application. Then close the Developer Studio Viewer before continuing the tutorial.

   **Note:** When you click Exit, you see the WFmstart.html page in the Developer Studio Viewer. This is the Developer Launch Console. For more information, see Running WebFOCUS Maintain Applications in the Developing WebFOCUS Maintain Applications manual.
Deploying Your Project to a Different WebFOCUS Server

Up until this point, you have been running your project locally. Now that the project is fairly complete, deploy it to another WebFOCUS Server.

You determine where your procedure is deployed using a *deployment scenario*. The deployment scenario maps all of the components in your project to a WebFOCUS Server or a web server. Then, at deployment time, WebFOCUS Maintain makes sure that all the pieces get updated in the appropriate locations.

By default, all WebFOCUS Maintain projects contain a Local Deploy scenario, which deploys your project to your local development server.

You can have more than one deployment scenario, so depending on which deployment scenario is active, your application can be deployed in several places.

You can deploy and run your project on a different remote WebFOCUS Server, if you have already set up WebFOCUS Developer Studio to access a remote WebFOCUS Server. For more information, see the *Creating Reporting Applications With Developer Studio* manual.

**Procedure: How to Create a Deployment Scenario**

1. Create a new deployment scenario by right-clicking the *FanClub* project in the Explorer window, clicking *Project Deployment*, and then clicking *New Scenario*.

2. In the New Scenario dialog box, give your new deployment scenario the name *Remote Deploy*. 
3. Click Next, as shown in the following image.

4. Enter *fanclub* as the target application name, as shown in the following image.
Note: If you choose to deploy to your development server, you must enter a name other than Fanclub, which is your development directory, as the target application name.

5. Click Next.
6. Select Start.mnt as your starting object, as shown in the following image.

7. Click Next.

8. Review the information and press Finish.

WebFOCUS Maintain adds a new folder, Deploy, to the list of folders under the project in the Explorer. Open the Deploy folder. You will see two deployment scenarios:

- Local Deploy is the default deployment scenario created when you created your project. It deploys your project locally on your machine. You cannot modify or delete this deployment scenario.
Remote Deploy is the deployment scenario you just created, as shown in the following image.

In the next sections, you will set up Remote Deploy to deploy to a remote WebFOCUS Server.

**Procedure: How to Set up a Deployment Scenario**

After you have set up a remote server, you must set up the deployment scenario to deploy to it.

1. In the FanClub project, open the Deploy folder.
2. Open **Remote Deploy**.

On the left side, you see the files that are in your project. Currently, you have two: fannames.mas and start.mnt.
On the right side, you see the servers to which you can deploy project files. In this example, there are two: localhost (your local WebFOCUS Server) and Remote Server, but you may have more. An example of this is shown in the following image.

3. Expand your remote server by clicking the + button next to its name.
   You should see at least two folders under its name. An example of this is shown in the following image.

   ![Remote Server](image)

   Folders that start with WEB represent servers at your target server. Folders that start with EDA represent WebFOCUS Servers at your target server. In general, you deploy files that you create with WebFOCUS Developer Studio (WebFOCUS files, Maintain files, Master Files) to the WebFOCUS Server, and you deploy third-party files (HTML files, images, script files) to the web server. There are exceptions to this rule. See *Partitioning and Deploying Project Files* in the *Creating Reporting Applications With Developer Studio* manual for more information.

4. Assume that fannames.mas already resides on your target web server. This means that you do not need to deploy fannames.mas, since it is already there.
Drag start.mnt to EDA/EDASERVE/APPPATH/FanClub, as shown in the following image.

Notice that the check mark icon ✔ appeared next to Start.mnt, and the Assigned attribute has changed to Yes.

5. You do need to tell the FanClub application where to find the fannames data source. Right-click the target server, and click Set Server Paths.

6. In the Set Running Paths for Server dialog box, add the directory where the fannames data source resides to the Running Paths box, as shown in the following image. If this server is similar to your installation, this file is likely to be in the Maintain directory. Otherwise, you can open the Remote Application Servers in the Explorer, and open the target server.
7. When you are done, click OK.

You are now ready to deploy and run your application using your brand new deployment scenario.

**Procedure: How to Deploy and Run Your Application Using Your New Deployment Scenario**

1. On the Exploring toolbar, make Remote Deploy your current deployment scenario by selecting it from the pull-down menu of deployment scenarios.

2. Click the **Deploy and run scenario** button, either in the Remote Deploy window, or on the Exploring toolbar.

WebFOCUS Maintain copies files to the remote server, while displaying the Deploying Application dialog box, as shown in the following image.

When it is done, click **Close** to run the application.

   **Note:** The fannames data source on the remote server may have different data from your local copy.

3. When you are done running your application, click **Exit** and then close the Developer Studio Viewer.

4. In the Maintain Development Environment, close the Remote Deploy deployment scenario.

Since WebFOCUS Developer Studio and WebFOCUS Maintain share their Deploy tool, more detailed information about deploying is available in the *Creating Reporting Applications With Developer Studio* manual.

**Adding Images to Your Project**

You can use images to improve the appearance and usability of WebFOCUS Maintain applications. This section explains how to:

- Add an image to the background of your two forms. This image, spiralbg.gif, makes your forms look like pages in a spiral notebook.

- Add titles to your two forms.

- Add a fan graphic to your form.

**Procedure: How to Add a New Background Image to Your Form**

1. Open Form1.

2. Make sure all of the controls are deselected so that you can see the properties for the form.
3. In the property sheet for the form, double-click the `BackgroundImage` property.
   WebFOCUS Maintain opens the Image Source dialog box, as shown in the following image.

4. Click `New`.
   WebFOCUS Maintain opens the Resource Wizard so that you can define this image as a resource in your project.

5. When the Resource Wizard asks you to specify the path to the image, click the ellipses button to use an Open box to find the image, select `spiralbg.gif`, and click the `Open` button, as shown in the following image.
6. Click Next.
7. In the final window of the Resource Wizard, you specify a name for your image, as shown in the following image. By default, WebFOCUS Maintain uses the name of the image.

8. Click Finish.

WebFOCUS Maintain displays a dialog box informing you that a copy of this file must be created in the project directory.

9. Click OK.
You return to the Image Source dialog box, where your image is now selected as the background image, as shown in the following image.

10. Click OK.
Your form resembles the following:

You may need to move the controls on your form to the right so that they are not overlapping the spiral. In the Edit menu, click Select all and move them.

Tiling Images

If you recall, Spiralbg was a short, wide image, but in your form it seems to take up the entire left side. Spiralbg is being tiled, that is, displaying multiple copies of itself. If you want a repetitive design on your form, it is better to use a small image and tile it instead of using a large image with the full picture you want. This conserves memory and ensures that the image looks good no matter how big or small the window of the end user is.

The Image Source dialog box automatically tiled Spiralbg by selecting 0 - Full in the Tiling style list.

Using Folders in the Explorer

WebFOCUS Maintain has now added this graphic to your project.
Create a Graphics folder to display all of the graphics you add to the project.

Now that WebFOCUS Maintain has added Spiralbg to the FanClub project, it now knows where to find this image and you can easily add it to other forms in your project. Add it to ShowFan.

**Procedure: How to View Graphics Folders**

1. In the Explorer, right-click FanClub, click New, and then click Virtual Folder in the submenu.
2. In the New Virtual Folder dialog box, type Graphics in the Name box.
3. Select GIF Image as the contents of this folder, as shown in the following image.

![New Virtual Folder Dialog Box](image)

4. Click OK.

WebFOCUS Maintain adds the Graphics folder to the folders under FanClub. If you open it, you see Spiralbg.

**Procedure: How to Add an Existing Background Image to Your Form**

1. Open the ShowFan form.
2. Make sure all objects on the form are deselected so that you can see the properties for the form.
3. In the property sheet for the form, double-click the BackgroundImage property.

WebFOCUS Maintain opens the Image Source dialog box.
4. Select Spiralbg from the list of available images.
5. Click OK.
6. If necessary, move the HTML table to the right.
7. Deploy and run your application to see how it looks.
8. Click the Exit button to close the application before continuing the tutorial.

**Procedure: How to Add an Image to Your Form**

1. Open Form1.
2. Select all the controls on your form by pressing CTRL+A and move them down so that you have roughly one inch at the top of your form.

   **Note:** Use your rulers to determine how much room to leave at the top. If your rulers are not visible, select Rulers from the View menu or click the Toggle rulers button on the Layout toolbar.

3. Select the Image control button
4. Draw a box in the empty space at the top of the form.
   WebFOCUS Maintain opens the Image Source dialog box.
5. Repeat the steps you followed in *How to Add a New Background Image to Your Form* on page 90 to add Addafan.gif to your form.
6. Add the Fan.gif image to your form to the right of the entry boxes. (Repeat steps 3 through 5.)
Your form resembles the following:

7. Open ShowFan and add the image Currfan.gif. (Repeat steps 3 through 5.)
8. Deploy and run your application to see how it looks.
9. When you are done, exit your application by clicking Exit and closing the Developer Studio Viewer.

What Is Next?

To learn more about WebFOCUS Maintain, see WebFOCUS Maintain Advanced Tutorial on page 99.
Welcome to the WebFOCUS Maintain Advanced Tutorial. In this topic, you are going to continue developing the WebFOCUS FanClub application that you created in the Basic Tutorial. This application will enable you to maintain a data source of WebFOCUS "fans." This tutorial assumes that you have already completed the WebFOCUS Maintain Basic Tutorial.

In this chapter:

- Before You Begin
- Creating an Update Form
- Updating a Record in a Data Source
- Using Scripts for User Feedback
- Updating a Data Source Using a Read/Write Grid
- Creating a Welcome Screen Using the Menu Control
- Adding a WebFOCUS Report to Your Application
- Using a JavaScript Function to Define an Email Link
- Adding a Pop-up Calendar
- Getting Help in the Maintain Development Environment

Before You Begin

Before you begin working on the WebFOCUS FanClub application, make sure you have done the following:

- Completed the WebFOCUS Maintain Basic Tutorial.

- Checked to see that you have the sample tutorial files required for the WebFOCUS Maintain advanced tutorial. This includes the following files:
  - The fanrpt.fex external procedure. You can find this file in the Ibi\Apps\Maintain folder.
  - The Showall.gif and WebFOCUS1.gif image files. You can find these images in the Ibi\Apps\Maintain\Images folder.
Creating an Update Form

If you want to skip around among the exercises in this chapter, you can do so, with the following exceptions:

- You must do Creating an Update Form on page 100 before Updating a Record in a Data Source on page 107.
- You must do Updating a Record in a Data Source on page 107 before Using Scripts for User Feedback on page 117.
- You must do Updating a Record in a Data Source on page 107 before Updating a Data Source Using a Read/Write Grid on page 118.
- You must do Creating a Welcome Screen Using the Menu Control on page 124 before Adding a WebFOCUS Report to Your Application on page 129.
- You must do Adding a WebFOCUS Report to Your Application on page 129 before Using a JavaScript Function to Define an Email Link on page 136.

Creating an Update Form

You can add fans to the WebFOCUS FanClub application, but what happens when you want to update or remove fans? You need to create a form where you can change information about a fan or remove a fan from the data source.

This section is divided into the following:

- Defining a link from the HTML Table in ShowFan that opens the Update/Delete form.
- Creating the Update/Delete form.
- Writing the code that updates the data source.
- Writing the code that deletes the record from the data source.
- Writing the code that deletes the record from GetFanStack.
Defining HTML Table Links

The end user will open the Update/Delete form by selecting a fan from the list of fans in the ShowFan form. As you will recall from the Basic Tutorial, ShowFan uses an HTML Table to display the contents of the data source stack GetFanStack, which contains all of the fans in the fannames data source. The following image illustrates the front end of the ShowFan and Update/Delete forms.

![HTML Table Illustration]

The application will determine which row the end user selected using the ClickRow syntax:

```
formname tablename.ClickRow
```

The header row returns 0, and the first data row returns 1.
In order to take advantage of this feature, you need to enable links for the HTML Table, which you do by opening the HTML Table Control Columns dialog box, and then opening the Table Column dialog box for the Last column.

You also need to use the special ClickLink event instead of the Click event. ClickLink is unique to HTML Tables.

The row that the end user clicks in the HTML Table corresponds to the row in GetFanStack that you want to display in the Update/Delete form. Each data source stack has an associated variable called FocIndex, which determines what the current row is. When the end user clicks a row in the HTML Table on ShowFan, the FanClub application needs to set FocIndex to whatever the value of the row is. It does this using the COMPUTE command. It then opens the Update/Delete form.

The syntax of the COMPUTE command is

```compute
variable = expression;
```

where:

- `variable` is the FocIndex of GetFanStack (written GetFanStack.FocIndex).
- `expression` is ShowFan.HTMLTable1.ClickRow.

For the complete syntax of COMPUTE, see Command Reference in the Maintain Language Reference manual.

**Procedure: How to Enable HTML Table Links**

1. If you have not already done so, open the WebFOCUS FanClub application in the Maintain Development Environment.
2. Create a new form in the Start procedure.
3. Give it the name Update_Form.
4. Give it the title Update or Remove a Fan.
5. Open the ShowFan form.
6. Double-click the HTML Table to open the HTML Table Control Columns dialog box.
7. Double-click the Last column to open the Table Column dialog box.
8. In the Links list, select *Body only*, as shown in the following image.

9. Click **OK** to close the Table Column dialog box.

10. Click **OK** to close the HTML Table Columns dialog box.

11. Right-click the HTML Table, and in the shortcut menu, click *Edit event handlers*.

12. Select the *ClickLink* event.

13. Enter the following syntax in the Event Handler editor:

   ```
   COMPUTE GetFanStack.FocIndex = ShowFan.HTMLTable1.ClickRow;
   Winform Show Update_Form;
   ```

   **Tip:** You can drag GetFanStack.FocIndex and Update_Form from the Project Explorer to the Event Handler editor to generate this syntax. You will have to expand GetFanStack in the Project Explorer to see FocIndex. You will have to type the rest of the syntax.

Creating the Update/Delete Form

Your next step is to place controls on Update_Form. This form should look almost exactly like Form1. Making your forms look similar this way makes it easier for end users to learn your application, because forms that perform similar tasks work the same way. In fact, before you start developing your own applications, developing a template for the application is recommended.

For more tips on developing forms, see Developing and Using Forms in the Developing WebFOCUS Maintain Applications manual.

Procedure: How to Add Controls to Update_Form

1. Make Update_Form the active form.
2. Make Spiralbg the background image.
3. Drag GetFanStack from the Project Explorer to Update_Form.

   WebFOCUS Maintain opens the Select Stack Columns dialog box, as shown in the following image.

4. Select SSN, LASTNAME, FIRSTNAME, COMPANY, ADDRESS, CITY, STATE, ZIP, PHONE, EMAIL, and ENROLLMENT_DATE (all of the columns except for the last two).
5. Click OK.

   WebFOCUS places prompted edit fields for all of these columns on Update_Form.
6. Change the prompt for Lastname to Last and the prompt for Firstname to First.
7. Move Last to the right, and move First up to its left.
8. Move up Company.
9. Move up Address.
10. Move up City.
11. Move State to the right of City.
12. Move Zip to the right of State.
13. Move up Phone.
14. Move up Email.
15. Move up Enrollment_Date.
16. Select Ssn, First, Company, Address, City, Phone, Email, and Enrollment_Date.
17. On the Layout toolbar, click Align Edges.

   Then click the Center vertically button.

   (This command does not actually center the controls. Instead it aligns the prompted edit boxes by the left sides of the edit boxes.)

18. If necessary, select all of the controls and move them down so that you have room to add a title to a form.
19. Select the Text control.
20. Draw a box at the top of the form approximately where you want your title to be.
21. The text in the text control should be selected. Type Update or Remove a Fan and press Enter.
22. Double-click the Font property for the text control to open the Font dialog box.
23. Change the Font style to Bold and the Size to 18, and click OK.
24. Create a button with the name UpdateButton and the text Update.

**Note:** You cannot name the Update button "Update" because Update is a reserved word. Reserved words are command names, built-in function names, keywords, and other WebFOCUS terms that could cause unpredictable behavior if you use them to name components of your application.

However, there is no reason the text on the button (what the end user sees) cannot say Update.
For a complete list of reserved words, see Language Rules Reference in the Maintain Language Reference manual.

25. Create a Delete button named DeleteButton. Delete is also a reserved word.

26. Make the Ssn field read-only. Select it, then select Ssn_Edit in the drop-down list of controls in the Property sheet. Change the ReadOnly property from 0 - No to 1 - Yes.

Copying Controls

Update_Form now looks almost like Form1 except that it has a slightly different purpose. However, Form1 has a group of radio buttons that enables the end user to supply the title of a fan (Mr., Mrs., or Ms.), and Update_Form needs this same set of radio buttons.

Rather than creating new radio buttons, you can copy them from Form1 and paste them into Update_Form.

WebFOCUS Maintain performs two types of copying; deep copy and shallow copy. When you perform a deep copy, WebFOCUS Maintain copies both the appearance of a control and any underlying data bindings, event handlers, and connections to resources. When you perform a shallow copy, WebFOCUS Maintain copies appearance only.

What happens if you perform a deep copy of the radio buttons from Form1 into Update_Form? You will see the three options (Mr., Mrs., and Ms.) and the tool tip text (Please select one) and the radio buttons will be bound to AddFanStack.TITLE (the TITLE column of AddFanStack). Since Update_Form uses GetFanStack, you need to change this binding to GetFanStack.TITLE. If you perform a shallow copy, you will see the three options and the tool tip text only. So, you do not save any time by using either a deep copy or a shallow copy.

You also need a Back button to close the form and return to the ShowFan form. The form ShowFan has a Back button, so you can copy it from here. If you perform a deep copy, you will get the button and the event handler self.WinClose(). If you perform a shallow copy, you will get the button only. You will have to reassign the event handler.

**Procedure:** How to Copy Controls to Update_Form

1. Open Form1, and copy the Mr., Mrs., Ms. set of radio buttons.

2. Close Form1 and paste the radio buttons into Update_Form by clicking Paste appearance in the Edit menu. (This command performs a shallow copy.)

3. Use the SelectedItem property to bind the set of radio buttons to GetFanStack.TITLE.

4. Open Show Fan and copy the Back button.
5. Close ShowFan and paste the Back button into Update_Form by clicking Paste in the Edit menu. (This command performs a deep copy.) You may need to reposition the button on your new form.

(If you wish, double-click the Back button to open the Event Handler editor to see that the self.WinClose (); syntax is still there.)

More About Copying Controls

You can always perform a shallow copy, but sometimes you cannot perform a deep copy. For example, if you copied the radio buttons from Form1 and tried to do a deep copy into another form in another procedure, you would no longer see the data binding to AddFanStack.TITLE, because AddFanStack is only defined in the context of the Start procedure. The new procedure does not know what AddFanStack is, so it loses the binding.

Updating a Record in a Data Source

Your next step is to create the code that will update the fan information for the fan shown in Update_Form. Generating this code using the Language Wizard is fairly straightforward.

Procedure: How to Update a Record in a Data Source

1. Create a new function in the Start procedure named UpdateFan.
2. In the Procedure Editor, open UpdateFan.
3. Make sure your cursor is between Case UpdateFan and EndCase and open the Language Wizard.
4. In the initial Language Wizard window, select Update records in a data source and click Next.
5. When the Language Wizard asks you how you would like to update records in a data source, select Update certain data source columns, one or more records at a time and click Next.
6. Expand the fannames data source to see the segments and fields.
7. In the list of available fields, select LASTNAME, FIRSTNAME, COMPANY, ADDRESS, CITY, STATE, ZIP, PHONE, EMAIL, TITLE, and ENROLLMENT_DATE (everything but SSN and USER).

You must select individual fields that you want to update in a data source. Unlike when you generated the ADD command using the Language Wizard. This gives you the flexibility to update just the fields that you want.

You cannot update SSN because it is a key field.
Notice that you can also select the `CUSTOMER` segment and move it over. The Language Wizard knows to move only updatable fields, as shown in the following image.

8. Click Next.

9. You now need to specify the record that will be updated in the `fannames` data source. This record corresponds to the current row of `GetFanStack`.

   Select the `Stack` radio button and select `GetFanStack` from the list of data source stacks in your procedure.

10. Remove the 1 in the Starting from row box and type in the word `FocIndex`. 
11. Select the *A specific number* radio button and make sure that 1 is entered in the box, as shown in the following image.

![Maintain Language Wizard](image)

12. Click *Finish*.

The Language Wizard generates the following code:

```sql
Update fannames.CUSTOMER.LASTNAME fannames.CUSTOMER.FIRSTNAME
fannames.CUSTOMER.COMPANY fannames.CUSTOMER.ADDRESS
fannames.CUSTOMER.CITY fannames.CUSTOMER.STATE fannames.CUSTOMER.ZIP
fannames.CUSTOMERPHONE fannames.CUSTOMER.EMAIL
fannames.CUSTOMER.TITLE fannames.CUSTOMER.ENROLLMENT_DATE
from GetFanStack(GetFanStack.FocIndex);
```

For more information on the UPDATE command, see the *Command Reference* in the *Maintain Language Reference* manual.

13. In Update_Form, double-click *UpdateButton* to open the Event Handler editor.

14. Assign the UpdateFan function to the Click event for UpdateButton.
Deleting a Record From a Data Source

Your next step is to create the code that will delete the fan shown in the Update/Delete form from the fannames data source. Generating this code using the Language Wizard is fairly straightforward.

However, think about the flow of forms in your application. After you delete the fan from the data source, you do not want to display the information about the fan in the Update_Form, so you should close the Update_Form.

Instead of closing using the Close form button in the Event Handler editor, you will use the Winform command to close the form. The Winform command enables you to manipulate forms at run time, including opening forms, closing forms, determining properties of forms and their controls, and setting properties of forms and controls.

Procedure: How to Delete a Record From a Data Source

1. Create a new function in the Start procedure named DeleteFan.
2. Open DeleteFan in the Procedure Editor.
3. Make sure your cursor is between Case DeleteFan, EndCase. Open the Language Wizard.
4. In the initial Language Wizard window, select Update records in a data source and click Next.
5. When the Language Wizard asks you how you would like to update records in a data source, select Delete one or more records from the data source and click Next.
6. Expand the fannames data source to see the segments and fields.
7. Select the fields in the CUSTOMER segment of the fannames data source or the CUSTOMER segment itself. Notice that when you select one field, the rest of the fields are also copied. This is because when you delete a record from a data source, you want to delete the contents of all of the fields associated with that record.
8. Click Next.
9. You now need to specify the record that will be deleted from the fannames data source. This record corresponds to the current row of GetFanStack.

   Select the Stack radio button and select GetFanStack from the list of data source stacks in your procedure.

10. Remove the 1 in the Starting from row box and type in the word FocIndex.
11. Select the *A specific number* radio button and make sure that 1 is entered in the box, as shown in the following image.

The Language Wizard generates the code

```
Delete fannames.CUSTOMER.SSN from GetFanStack(GetFanStack.FocIndex);
```

which means, Delete the record that corresponds to the current row of GetFanStack from the fannames data source. For more information on the DELETE command, see the *Command Reference* in the *Maintain Language Reference* manual.

12. Click *Finish*.

13. You now need to insert the code that will close Update_Form. Make sure your cursor is between the code you just generated and EndCase and open the Language Wizard again.

14. In the initial Language Wizard window, select *Operate on a form* and click *Next*.

15. When the Language Wizard asks you what form operation you would like to perform, select *Close a form* and click *Next*. 
16. When the Language Wizard asks you to select a form to close, select Update_Form and click Finish.

The Language Wizard generates the following syntax:

```
Winform Close Update_Form;
```

For more information on the Winform command, see the Command Reference in the Maintain Language Reference manual.

17. In Update_Form, double-click DeleteButton to open the Event Handler editor, and assign DeleteFan to the Click event for DeleteButton.

**Deleting a Record From a Data Source Stack**

After you close Update_Form, you return to the ShowFan form, which contains an HTML Table that displays the contents of GetFanStack. Unless you do something, the fan that you just deleted from the fannames data source will still be in GetFanStack and visible to the end user.

So, in addition to deleting the fan from fannames, you also have to delete the fan from GetFanStack.

Your first inclination might be to reload the data from fannames into GetFanStack. However, data source operations such as this one are often the bottlenecks in application design. Your application will run much more quickly if you simply delete the row from the stack, especially if your data source has a lot of records or resides on another computer.

Take a look at how to delete a row from a stack. In the following figure, you can see a diagram of what a stack looks like. The current row, the one you want to delete, is numbered FocIndex, and the number of rows in GetFanStack is FocCount.
You can leave the rows from 1 to FocIndex–1 alone, as shown in the following image.

The rows from FocIndex+1 to FocCount each need to move up one. To do this, you will copy the contents of the FocIndex+1 row into the FocIndex row. This discards the data in the FocIndex row, but that is fine because you wanted to delete it. You also have two copies of the data in the FocIndex+1 row.

Next, copy the contents of the FocIndex+2 row into the FocIndex+1 row, thus overwriting the original copy of the data in the FocIndex+1 row (but you now have the copy in the FocIndex row). You also have two copies of the data in the FocIndex+2 row.

You continue copying rows until you reach the end of the stack and have two copies of the data in the FocCount, or last row. At this point, you redefine the FocCount of the stack to be FocCount–1, thus discarding the extra copy of the data in the FocCount row.
There is a special case when FocIndex is the last row of the stack. In this case, all you need to do is redefine both FocIndex and FocCount to be FocIndex–1 and FocCount–1, respectively, as shown in the following image.

![Diagram showing the redefinition of FocIndex and FocCount]

**Procedure:** How to Delete a Row From a Stack

1. Create a variable in the Start procedure by right-clicking Start, clicking New in the shortcut menu, and then clicking Variable (Declare).

2. In the New Variable dialog box, give your variable the name CNT.

3. Click the ellipsis button to open the Type Wizard. (The Type Wizard helps you assign a data type to your new column.)
4. Leave the Built-in Type and Simple and Integer choices, but change the size to 2, as shown in the following image.

Notice at the bottom of the dialog box, that the Type Wizard uses an abbreviation of I2 to represent your data type choice.

5. Click OK to return to the New Variable dialog box.
Notice that the Type Wizard has transferred the I2 to the Type box. If you know the WebFOCUS abbreviation for your data type, you can type it directly here, as shown in the following image. You will also notice, as you continue to work in a project, that WebFOCUS Maintain saves the data types you have applied and lists them.

6. Click OK to close the New Variable dialog box.

**Note:** If you double-click the CNT variable, you will see your new CNT variable listed under the $$Declarations line at the top of the Start procedure under the MAINTAIN command line.

7. Open the DeleteFan function.

8. Place your cursor between the line with the DELETE command and the line with the Winform command. This placement is important, because you want to delete the record from GetFanStack after you delete it from fannames, but before you redisplay GetFanStack.

9. Enter the following code:

   ```webfocus
   IF GetFanStack.FocCount=GetFanStack.FocIndex THEN
   BEGIN
   COMPUTE GetFanStack.FocCount = GetFanStack.FocCount-1;
   COMPUTE GetFanStack.FocIndex = GetFanStack.FocIndex-1;
   ENDBEGIN
   ELSE
   BEGIN
   COMPUTE CNT = GetFanStack.FocIndex;
   Stack copy from GetFanStack(CNT+1) into GetFanStack(CNT) ;
   COMPUTE GetFanStack.FocCount = GetFanStack.FocCount-1;
   ENDBEGIN
   ```
**Using Scripts for User Feedback**

In a previous section, you added code to your FanClub application that deletes the name of a fan from the fannames data source.

However, you did not add any code asking your end users whether they are really sure about deleting this fan name.

You could easily create a confirmation form that asked your end users, "Are you sure you want to delete this fan?" If they click Yes, you execute the code to delete the fan. If they click Cancel, you return to Update_Form with the data source unchanged.

However, this means your application must reconnect to the WebFOCUS Server to get the confirmation form, which has the potential to slow down your application.

Instead of using Maintain forms and the Maintain language to create this confirmation form, you are going to use JavaScript. JavaScript (as well as VBScript) is a mini-scripting language that is ideal for validating actions locally.

**Note:** You must have completed *Creating an Update Form* on page 100 before you can do this section.
Using JavaScript to Confirm a Deletion

What does the JavaScript function have to do?

1. It has to ask end users whether they are sure about deleting the fan from the fannames data source.
   
   You do this using the JavaScript confirm function.

2. If the end user answers Yes, the JavaScript function needs to perform the Maintain function DeleteFan. If the end user answers Cancel, the function simply returns to Update_Form.
   
   You test what the end user response was using an if test. You execute DeleteFan using the IWCTrigger function, an Information Builders-supplied function that executes Maintain code from JavaScript.

Procedure: How to Confirm a Deletion Using JavaScript

1. Open the Event Handler editor for Update_Form and select the Click event for DeleteButton.

2. Delete the code in the Event Handler editor.

3. Click the JavaScript button.

4. Enter the following syntax:

   ```javascript
   if (confirm("Are you sure you want to delete this fan?"))
     IWCTrigger("DeleteFan");
   else
     alert("Delete cancelled!");
   ```

   **Note:** JavaScript is case-sensitive. Make sure you type this syntax exactly as shown.

5. Close the Event Handler editor, deploy and run your application.

Updating a Data Source Using a Read/Write Grid

Another option for updating the fannames data source is to create a form that includes a read/write grid. A read/write grid enables you to display the contents of a data source stack in a grid and to make changes directly to the stack. There are two read/write grids available in the MDE – an ActiveX Grid and a JS (JavaScript) Grid. This tutorial uses the ActiveX Grid, however, most of the information given here also applies to the JS Grid. For more information about the two read/write grids, see the Developing and Using Controls chapter in the Developing WebFOCUS Maintain Applications manual.
Procedure: How to Create An Update Form With a Read/Write Grid

The following procedure enables you to create a form containing a read/write grid.

1. Create a new form in the Start procedure.
2. Give it the name Grid_Form.
3. Give it the title Update Fans Using a Grid.
4. Open the ShowFan form.
5. Put a new button on the form called Grid Update Screen.
6. Double-click the Grid Update Screen button to open the Event Handler editor.
7. Click Event.
8. Drag the Grid_Form into the Event Handler editor and select Show a Form from the pop-up menu.

9. Reopen Grid_Form (if necessary) and on the Controls palette, click the Grid button.
10. Draw a rectangle on the form representing roughly where you want your grid to go on your form.

   WebFOCUS Maintain opens the Control Columns dialog box, where you define the contents of your grid.

11. Select GetFanStack from the Stack list.

    Note: Make sure you did not select AddFanStack.

    The Control Columns dialog box now displays the columns in the stack GetFanStack.

12. Copy the fields LASTNAME, FIRSTNAME, COMPANY, EMAIL, and TITLE into the Column selection list.
13. Use the Move up button and the Move down button to rearrange these fields so that they are in the following order: TITLE, FIRSTNAME, LASTNAME, COMPANY, and EMAIL, as shown in the following image.

You can change the appearance of any of these columns by opening the Grid Column dialog box. For example, suppose you want to change the header titles for the FIRSTNAME and LASTNAME fields so that they read First and Last.

15. In the Grid Column dialog box, change the Header title to First, as shown in the following image.

You can also change the width, justification, font, and color if you wish.

16. Click OK.

17. Repeat the process to change Lastname to Last.

18. Click OK to leave the Column Properties dialog box, and click OK to close the Grid Control Columns box.
19. Copy navigation buttons onto the Grid_Form.
   - Open Update_Form and copy the Update button.
   - Close Update_Form and paste the Update button into Grid_Form by clicking Paste in the Edit menu. (This command performs a deep copy.)
   - Open Update_Form and copy the Back button, then paste the Back button into the Grid_Form. You may need to adjust the position of these buttons on your form.

20. You will need to modify the logic behind the update button in the next procedure to allow you to update multiple fields at one time.

21. Copy the title (Update or Remove a Fan) from Update_Form and paste it onto Grid_Form.

22. Change the name to Update Fans Using a Grid.

23. Add spiralbg image to the background.

Your form will resemble the following:

![Image of Grid_Form with Update Fans Using a Grid title](image)

**Procedure:** How to Update Multiple Records Using the Read/Write Grid

When you copied the UpdateButton from the Update_Form, the deep copy imported the logic from the UpdateFan case. This logic updates only one field in a stack at a time. The following modification to the logic behind the Update button enables you to update the entire GetFanStack at once.
1. Double-click UpdateFan case in the Project Explorer to access the following code:

```sql
Case UpdateFan

Update fannames.CUSTOMER.LASTNAME fannames.CUSTOMER.FIRSTNAME
fannames.CUSTOMER.COMPANY fannames.CUSTOMER.ADDRESS
fannames.CUSTOMER.CITY fannames.CUSTOMER.STATE fannames.CUSTOMER.ZIP
fannames.CUSTOMERPHONE fannames.CUSTOMER.EMAIL
fannames.CUSTOMER.TITLE from GetFanStack(GetFanStack.FocIndex) ;

EndCase
```

2. Highlight the block of code beginning with Case UpdateFan and ending with EndCase.

3. Copy and paste the code beneath EndCase.

4. Rename the pasted code Case GridUpdate. This creates a new function.

5. Replace Update fannames with For all Update fannames. This enables you to update all
   fields in the data source stack.

6. Replace GetFanStack(GetFanStack.FocIndex) with GetFanStack(1).

7. When you have modified the code, it should appear as the following:

```sql
Case GridUpdate

For all Update fannames.CUSTOMER.LASTNAME fannames.CUSTOMER.FIRSTNAME
fannames.CUSTOMER.COMPANY fannames.CUSTOMER.ADDRESS
fannames.CUSTOMER.CITY fannames.CUSTOMER.STATE fannames.CUSTOMER.ZIP
fannames.CUSTOMERPHONE fannames.CUSTOMER.EMAIL
fannames.CUSTOMER.TITLE from GetFanStack(1);

EndCase
```

8. Save your changes. The GridUpdate case appears in the Project Explorer after you have saved
   the new code.

9. Double-click the Update button on the Grid_Form to open the Event Handler editor.

10. Change Perform UpdateFan( ); to read Perform GridUpdate( );.

11. Close the Event Handler editor.

The Update button on the Grid_Update form will now update the entire fannames data source,
allowing you to make multiple changes to the read/write grid and update with one click.
Using a Read/Write Grid

Try running your application and accessing the Grid_Form from the ShowFan form. Change some of the values in the grid and click the Update button. You should see the changes you made in the grid appear on ShowForm. Note that using the read/write grid to update extremely large data sources is not recommended. For detailed information on grid functionality, see Developing and Using Controls in the Developing WebFOCUS Maintain Applications manual.

Creating a Welcome Screen Using the Menu Control

Instead of bringing your end users directly into your FanClub application, it is nice to display a welcoming screen that gives them an idea of the kinds of things your application will do.

In this section, you will create such a screen. You will also learn how to use the Maintain Development Environment menu control to place a menu bar on your form.

Procedure: How to Create a Welcome Screen

1. Create a new form in the Start procedure.
2. Name the form Welcome_Form, and give it the title Welcome to the WebFOCUS Fan Club.
3. Make the background image Spiralbg.
4. Click the Text control and draw a rectangle at the top of your form.
5. Type the text, Welcome to the FanClub Application.
6. In the properties tab, specify the font as Times New Roman, Italic, 22 point. Specify the ForeColor as Red.
7. Add the image webfocus1.gif to the bottom of your form, as shown in the following image.

![Welcome Screen Image](image)

This provides a basic Welcome Screen to greet the user, but it does not give the user much to do. To provide the user with a more meaningful welcome screen, you can create a menu bar that will allow your user to take advantage of your entire application.

**Procedure: How to Create a Menu Bar**

1. From the Controls Palette, select the *Menu* control.
2. Draw a rectangle on your form in order to create a menu bar.
3. Name the menu bar WelcomeMenu.
4. In the Properties tab, change Orientation to Vertical. Because you will be specifying fairly long menu options, a vertical menu bar will be more practical.
5. Double-click the menu bar to open the Menu Items dialog box.
6. Create three more menu items on your form by selecting WelcomeMenu and clicking the *New* button three times.
7. Change the text of the menu items as follows:

- Menu1 to Add a Fan
- Menu2 to Browse a Fan
- Menu3 to Show All Fans
- Menu4 to Exit

Notice that when you type in the Text box, WebFOCUS Maintain echoes what you type in the Name box, minus the spaces. Leave the names as WebFOCUS Maintain types them. An example of this is shown in the following image.

8. Press Enter or click OK when you have finished adding all of the menu items.

9. In the Properties tab, change the properties for the WelcomeMenu control. For example:

- Set BackColor to blue.
- Set ForeColor to white.
- Set Font to Times New Roman, Italic, 16pt.
- Set BackColorOver to light blue.
- Set ForeColorOver to yellow.
Assigning Event Handlers to a Menu Item

After you place a menu bar on your form, WebFOCUS creates click events that correspond to menu items. Each event is named ClickMenuItem, where MenuItem is the name of each menu item you created.

If you select the WelcomeMenu and select the Events tab in the Property sheet, you will see the four events that correspond to the items you specified for your menu, as shown in the following image.

You can now assign event handlers to these events.
**Procedure:** How to Assign Event Handlers to a Menu Item

1. Select the *WelcomeMenu* menu bar, then in the Property sheet, select the *Events* tab.
2. Double-click the *ClickAddaFan* event to open the Event Handler editor.
3. Assign Form1 to this event by dragging Form1 from the Project Explorer into the Event Handler. Select *Show a Form* from the pop-up menu.
4. In the Events tab, double-click the *ClickBrowseaFan* event.
5. Assign the function GetFans and the form ShowFan to this event (drag them both from the Project Explorer as well).
6. Double-click *Click Exit* in the Events tab.
7. The Exit option should close the application. Assign *Close project* to this event. (You are not going to assign ShowAllFans in this section.)
8. Close the Event Handler editor and save your changes.

**Adjusting the Application Flow of Forms**

Before you can deploy and run the FanClub application, you must adjust the navigation of forms in the application. Currently, Form1 is the first form displayed to the end user. You must change this to *Welcome_Form*. Also, Form1 has an Exit button that closes the application.

**Procedure:** How to Adjust the Application Flow of Forms

Change this to a Back button that will just close Form1.

1. In the Project Explorer, open the Top function. This actually opens the entire Start procedure in the Procedure Editor with your cursor in the Top function.
2. Delete the line
   
   `Winform Show FORM1;`

3. Use the Language Wizard to add the following code in its place:
   
   `Winform Show WELCOME_FORM;`

4. Close the Procedure Editor.

5. Open Form1.

6. Select *ExitButton* and change its name to *BackButton*.

7. Change the text from Exit to Back.
8. Open the Event Handler editor, delete the existing event handler for Click, and assign the event to Close form.


10. Delete the Show Fans button.

11. Deploy and run your application.

12. When you are done, close your application by clicking Exit and closing the Developer Studio Viewer.

Adding a WebFOCUS Report to Your Application

In the previous section, you added a welcome screen to your application with four menu options. One of these options, Show all Fans, does not do anything. You are going to add another form to your application that displays the results of a WebFOCUS report. This report procedure, fanrpt.fex, is one of the sample files that came with WebFOCUS. The front end of the Show all Fans menu item is shown in the following image.

![WebFOCUS Report](image)
You can use the results of a WebFOCUS procedure in your application in one of two ways:

- You can return the formatted results of the report into a one-column stack, which you then display using an HTML Object.
  
The stack can be named anything, but the column in the report must be named HTML.
  
The report must have the line:
  
  ON TABLE PCHOLD FORMAT HTMTABLE

- You can return the unformatted columns of the report into a stack with columns that have the same name as the columns in the report.
  
The report must have the line:
  
  ON TABLE PCHOLD

**Procedure:** How to Add a WebFOCUS Report to Your Application

1. In the Explorer, open the Procedures folder.
2. Make sure that Display all files is turned on. Either in the View menu, or by clicking the button on the Exploring toolbar.
3. Right-click *fanrpt.fex* and click Add to project.

**Procedure:** How to Define a Stack to Receive WebFOCUS Report Output

1. Create a new data source stack in the Start procedure by right-clicking Start and clicking New data source stack.
2. Give your stack the name HtmlStack.
   
   Up until this point, you have created data source stacks based on the fields of a data source. For this stack, you will create a custom column.
3. Double-click New variable to open the Computed Stack Column dialog box.
4. Give your column the name HTML (if you name it anything else, the report will not work), as shown in the following image.

5. Click the ellipsis button ⋯ to open the Type Wizard. The Type Wizard helps you assign a data type to your new column.
6. Leave the Built-in Type and Simple choices, but change Integer to Alphanumeric and the size to 250, as shown in the following image.

![Type Wizard dialog box](image)

Notice at the bottom of the dialog box, that the Type Wizard uses an abbreviation of A250 to represent your data type choice.

7. Click **OK** to return to the Computed Stack Column dialog box.

   Notice that the Type Wizard has transferred the A250 to the Type box. If you know the WebFOCUS abbreviation for your data type, you can type it directly here, as shown in the following image. You will also notice, as you continue to work in a project, that WebFOCUS Maintain saves the data types you have applied and lists them.

![Computed Stack Column dialog box](image)

8. Click **OK** to return to the Stack Editor.

9. Click **OK** to confirm your new stack and close the Stack Editor.
**Procedure:** How to Execute Report Output Into a Stack

1. Create a new function in Start named ShowAllFans.
2. Double-click ShowAllFans to open it in the Procedure Editor.
3. Type the following text between Case ShowAllFans and EndCase:
   ```sql
   EXEC fanrpt INTO HTMLSTACK;
   ```
   Executing the ShowAllFans function runs the report and returns the results to HtmlStack.
4. Close the Procedure Editor.

**Procedure:** How to Display Report Output in an HTML Object

1. Create a new form to show the report output.
2. Name the form Show_Fans_Form.
3. Give the form the title Show All Fans.
4. Give it the Spiralbg background image.
5. Place the image Showall.gif at the top of the form.
6. Put a Back button on the form that closes the form.
7. Add an HTML Object to your form. In the Control palette, select the HTML Object control, and draw a box on your form.
8. Double-click it to open the HTML Content Source dialog box.
   The HTML Content Source dialog box enables you to specify HTML code that determines what the HTML Object displays. You could enter it directly here, but you want to use the report output from fanrpt.
9. Change Insert HTML content from As entered here to From a variable.
10. Expand HtmlStack.
11. Select HTML, as shown in the following image.

12. Click OK to close the HTML Content Source dialog box.

13. Change the Overflow property for the HTML Object to 2 - Scroll.

14. Close Show_Fans_Form.

15. In the Project Explorer, right-click Welcome_Form, and in the shortcut menu, click *Edit event handlers* to open the Event Handler editor.

16. In the list of controls, select *WelcomeMenu*.

17. In the list of events, select *ClickShowAllFans*.

18. Drag ShowAllFans from the Project Explorer into the Event Handler editor.
19. Drag Show_Fans_Form from the Project Explorer into the Event Handler editor and select Show a Form from the pop-up menu, as shown in the following image. (Be sure to select Show_Fans_Form and not ShowFan.)

20. Close the Event Handler editor.

21. Deploy and run your application.

**Clearing HtmlStack**

If you run your application, and enter and leave Show_Fans_Form multiple times in one session, you will notice that Show_Fans_Form displays multiple copies of the report.

Each time you enter Show_Fans_Form, WebFOCUS Maintain re-executes the report and places it in HtmlStack.HTML. This is Maintain language shorthand for the HTML column of the stack HtmlStack. However, since you did not clear the contents of HtmlStack.HTML before re-executing the report, the new report results get appended to whatever is already in the stack. You must clear HtmlStack before executing the report.

**Procedure: How to Clear HtmlStack**

1. Open Start in the Procedure Editor.
2. Find Case ShowAllFans.
3. Before the line that executes the report (Exec fanrpt INTO HTMLSTACK;), enter the following code:

   Stack clear HtmlStack ;

   You can either type this code or use the Language Wizard to enter it.
4. Close Start, save your project, and deploy and run it.
5. Try entering Show_Fans_Form more than once. The stack is clear.

Using a JavaScript Function to Define an Email Link

When you looked at your WebFOCUS report, you may have noticed that the EMAIL column contained links, as shown in the following image. Clicking on these links may result in a warning message.

The creator of this report set up the last column so that clicking it executes a script.

If you open the report (by double-clicking it) and page down almost to the bottom, you will see the following line of code:

```
TYPE=DATA, COLUMN=N4, JAVASCRIPT=Email_Script(N4), $
```

This line is looking for a JavaScript function named Email_Script. In this section, you will create this JavaScript function. The function will enable end users to send email to the address they click in this column.

**Procedure: How to Define Email_Script**

1. In the Explorer, right-click FanClub, point to New, and then click Project file.
2. From the Select type of the new item drop-down menu, select Java Script.
3. Give your JavaScript the name `MyScript`, as shown in the following image.

![Image of Create new project file dialog box with JavaScript selected, location set to IBFS/localhost/DEV/maintain, and ten name set to MyScript]

This command creates a JavaScript library. You will be writing the Email_Script function in this library.

4. Enter the following code:

```javascript
function Email_Script(address) {
    location = "mailto:" + address;
}
```

**Note:** Type this code exactly as shown. JavaScript is case-sensitive.

5. Close MyScript.
6. Open Show_Fans_Form.
7. Tile your screen so you can see both Show_Fans_form and the Explorer. Open the Other folder to display your JavaScripts.
8. Drag `MyScript.js` onto the ShowFan form. You must drop MyScript into an empty area on the form, not on top of a control.
9. You are prompted to either Embed or Link the Script. Select *Embed*.
10. Deploy and run your application.
Adding a Pop-up Calendar

You can use a pop-up calendar to populate and update the Enrollment_Date on the screens. You will add this to both the Add a Fan form (Form1) and the Update_form.

Adding a Pop-up Calendar to Your Forms

To do this, click each form and set the Calendar property in the Properties sheet to 1-Yes. This will provide a calendar icon at run time for your date-formatted edit box. By clicking the icon, a pop-up interactive calendar displays so that date values can be selected, rather than typed in. Save and run the file. The front-end result should look similar to the following image.

Getting Help in the Maintain Development Environment

The Maintain Development Environment enables you to get help quickly and easily using its Help file. There are three methods for getting help:
Open the Maintain Development Environment help file by clicking Contents and Index in the Help menu.

or

Click the Help button on the Miscellaneous toolbars.

You can also open the help file directly in the Language Reference section by clicking Language Reference in the Help menu.

Press F1 or click the Help button in any dialog box to see context-specific help about that dialog box.

In the Procedure Editor, select any keyword and press F1 to see help for that keyword.

You can also view a Tip of the Day by clicking Tip of the day in the Help menu or by clicking the Tip of the day button on the Miscellaneous toolbar.

These tips contain useful information about things you can do with WebFOCUS Maintain.

You can also view other files for WebFOCUS Developer Studio by opening them from the Start menu.
Procedure: How to View Context-Sensitive Help for a Dialog Box

1. Right-click the AddFan function, and in the shortcut menu, click Edit to open the Edit Function dialog box, as shown in the following image.

2. Click the Help button or press F1.

Procedure: How to View Maintain Language Syntax Help

1. Open the Start procedure in the Procedure Editor.
2. Select the MAINTAIN keyword.

The WebFOCUS Maintain online Help file opens at the relevant topic. If there is more than one relevant topic in the help file, you will see a list of them.
The following topics provide an overview of basic WebFOCUS Maintain concepts. To fully exploit the potential and productivity of WebFOCUS Maintain, you should become familiar with concepts including:

- Processing data in sets by using stacks.
- Controlling the flow of a WebFOCUS Maintain project and the procedures within it.
- Developing presentation logic using forms and event handlers.
- Reading from and writing to data sources.
- Ensuring transaction integrity.
- Creating classes and objects.

**In this chapter:**

- Set-based Processing
- Controlling the Flow of a Procedure
- Executing Other Maintain Procedures
- Forms and Event-driven Processing
- Reading From a Data Source
- Writing to a Data Source
- Transaction Processing
- Classes and Objects

### Set-based Processing

Maintain provides the power of set-based processing, enabling you to read, manipulate, and write groups of records at a time. You manipulate these sets of data using a data structure called a *data source stack*. 
A data source stack is a simple temporary table. Generally, columns in a data source stack correspond to data source fields, and rows correspond to records, or path instances, in that data source. You can also create your own user-defined columns.

The intersection of a row and a column is called a cell and corresponds to an individual field value. The data source stack itself represents a data source path.

For example, consider the following Maintain command:

```sql
FOR ALL NEXT Emp_ID Pay_Date Ded_Amt INTO PayStack
    WHERE Employee.Emp_ID EQ SelectedEmpID;
```

This command retrieves Emp_ID and the other root segment fields, as well as the Pay_Date, Gross, Ded_Code, and Ded_Amt fields from the Employee data source and holds them in a data source stack named PayStack. Because the command specifies FOR ALL, it retrieves all of the records at the same time. You do not need to repeat the command in a loop. Because it specifies WHERE, it retrieves only the records you need, in this case, the payment records for the currently-selected employee.

You could just as easily limit the retrieval to a sequence of data source records, such as the first six payment records that satisfy your selection condition

```sql
FOR 6 NEXT Emp_ID Pay_Date Ded_Amt INTO PayStack
    WHERE Employee.Emp_ID EQ SelectedEmpID;
```

or even restrict the retrieval to employees in the MIS department earning salaries above a certain amount:

```sql
FOR ALL NEXT Emp_ID Pay_Date Ded_Amt INTO PayStack
    WHERE (Employee.Department EQ 'MIS') AND
        (Employee.Curr_Sal GT 23000);  
```

Which Processes Are Set-based?

You can use set-based processing for the following types of operations:

- **Selecting records.** You can select a group of data source records at one time using the NEXT command with the FOR prefix. Maintain retrieves all of the data source records that satisfy the conditions you specified in the command and then automatically puts them into the data source stack that you specified.

- **Collecting transaction values.** You can use forms to display, edit, and enter values for groups of rows. The rows are retrieved from a data source stack, displayed in the form, and are placed back into a stack when the user is finished. You can also use the NEXT command to read values from a transaction file into a stack.
Writing transactions to the data source. You can include, update, or delete a group of records at one time using the INCLUDE, UPDATE, REVISE, or DELETE commands with the FOR prefix. The records come from the data source stack that you specify in the command.

Manipulating stacks. You can copy a set of records from one data source stack to another and sort the records within a stack.

The following diagram illustrates how these operations function together in a procedure:

The diagram is explained in detail below:

1. The procedure selects several records from the data source and, for each record, copies the values for fields A, B, and C into the data source stack. It accomplishes this using the NEXT command.

2. The procedure displays a form on the screen. The form shows multiple instances of fields A, B, and C. The field values shown on the screen are taken from the stack. This is accomplished using a form and the Winform Show command.

3. The procedure user views the form and enters and edits data. As the form responds to the activity of the user, it automatically communicates with the procedure and updates the stack with the new data.

4. The procedure user clicks a button to exit the form. The button accomplishes this by triggering the Winform Close command.

5. The procedure writes the values for fields A, B, and C from the stack to the selected records in the data source. The procedure accomplishes this using the UPDATE command.
How Does Maintain Process Data in Sets?

Maintain processes data in sets using two features:

- **The command prefix FOR.** When you specify FOR at the beginning of the NEXT, INCLUDE, UPDATE, REVISE, and DELETE commands, the command works on a group of records, instead of just one record.

- **Stacks.** You use a data source stack to hold the data from a group of data source or transaction records. For example, a stack can hold the set of records that are output from one command (such as NEXT or Winform) and provide them as input to another command (such as UPDATE). This enables you to manipulate the data as a group.

Creating and Defining Data Source Stacks: An Overview

Maintain makes working with stacks easy by enabling you to create and define a data source stack dynamically, simply by using it. For example, when you specify a particular stack as the destination stack for a data source retrieval operation, that stack is defined as including all of the fields in all of the segments referred to by the command. Consider the following NEXT command, which retrieves data from the VideoTrk data source into the stack named VideoTapeStack:

```
FOR ALL NEXT CustID INTO VideoTapeStack;
```

Because the command refers to the CustID field in the Cust segment, all of the fields in the Cust segment (from CustID through Zip) are included as columns in the stack. Every record retrieved from the data source is written as a row in the stack.

**Example:** Creating and Populating a Simple Data Source Stack

If you are working with the VideoTrk data source, and you want to create a data source stack containing the ID and name of all customers whose membership expired after June 24, 1992, you could issue the following NEXT command:

```
FOR ALL NEXT CustID INTO CustNames WHERE ExpDate GT 920624;
```

The command does the following:

1. Selects (NEXT) all VideoTrk records (FOR ALL) that satisfy the membership condition (WHERE).
2. Copies all of the fields from the Cust segment (referenced by the CustID field) from the selected data source records into the CustNames stack (INTO).

The resulting CustNames stack looks like this (some intervening columns have been omitted to save space):
Creating a Data Source Stack

You create a data source stack:

- Implicitly, by specifying it in a NEXT or MATCH command as the destination (INTO) stack, or by associating it in the Form Editor with a control.

  Forms are introduced in *Forms and Event-driven Processing* on page 164. The Form Editor used to design and create forms is described in Using the Form Editor in the *Developing WebFOCUS Maintain Applications* manual.

- Explicitly, by declaring it in an INFER command.

  For example, this NEXT command creates the EmpAddress stack:

  ```
  FOR ALL NEXT StreetNo INTO EmpAddress;
  ```

Defining Data Source Columns in a Data Source Stack

When you define a data source stack, you can include any field along a data source path. Maintain defines the data source columns of a stack by performing the following steps:

1. Scanning the procedure to identify all the NEXT, MATCH, and INFER commands that use the stack as a destination and all the controls that use the stack as a source or destination.
Set-based Processing

2. Identifying the data source fields that these commands and controls move in or out of the stack:
   - **NEXT commands.** Moves the fields in the data source field list and WHERE phrase.
   - **MATCH commands.** Moves the fields in the data source field list.
   - **INFER commands.** Moves all the fields specified by the command.
   - **Controls.** Moves all the fields specified by the control.

3. Identifying the data source path that contains these fields.

4. Defining the stack to include columns corresponding to all the fields in this path.
   - **NEXT commands.** Moves the fields in the data source field list and WHERE phrase.
   - **MATCH commands.** Moves the fields in the data source field list.
   - **INFER commands.** Moves all the fields specified by the command.
   - **Controls.** Moves all the fields specified by the control.

You can include any number of segments in a stack, as long as they all come from the same path. When determining a path, unique segments are interpreted as part of the parent segment. The path can extend through several data sources that have been joined together. Maintain supports joins that are defined in a Master File. For information about defining joins in a Master File, see the *Describing Data With WebFOCUS Language* manual. (Maintain can read from joined data sources, but cannot write to them.)

The highest specified segment is known as the anchor and the lowest specified segment is known as the target. Maintain creates the stack with all of the segments needed to trace the path from the root segment to the target segment:

- It automatically includes all fields from all of the segments in the path that begins with the anchor and continues to the target.
- If the anchor is not the root segment, it automatically includes the key fields from the anchor’s ancestor segments.
**Example:** Defining Data Source Columns in a Data Source Stack

In the following source code, a NEXT command refers to a field (Last_Name) in the EmplInfo segment of the Employee data source, and reads that data into EmpStack; another NEXT command refers to a field (Salary) in the PayInfo segment of Employee and also reads that data into EmpStack.

```
NEXT Last_Name INTO EmpStack;
.
.
FOR ALL NEXT Salary INTO EmpStack;
```

Based on these two NEXT commands, Maintain defines a stack named EmpStack, and defines it as having columns corresponding to all of the fields in the EmplInfo and PayInfo segments:

```
<table>
<thead>
<tr>
<th>Emp_ID</th>
<th>Last_Name</th>
<th>...</th>
<th>Ed_Hrs</th>
<th>Dat_Inc</th>
<th>...</th>
<th>Salary</th>
<th>JobCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>071382660</td>
<td>STEVENS</td>
<td>...</td>
<td>25.00</td>
<td>82/01/01</td>
<td>...</td>
<td>$11,000.00</td>
<td>A07</td>
</tr>
<tr>
<td>071382660</td>
<td>STEVENS</td>
<td>...</td>
<td>25.00</td>
<td>81/01/01</td>
<td>...</td>
<td>$10,000.00</td>
<td>A07</td>
</tr>
</tbody>
</table>
```

**Example:** Establishing a Path Using Keys and Anchor and Target Segments

The following code populates CustMovies, a data source stack that contains video rental information for a given customer. The first NEXT command identifies the customer. The second NEXT command selects a field (TransDate) from the second segment and a field (Title) from the bottom segment of a path that runs through the joined VideoTrk and Movies data sources:

```
NEXT CustID WHERE CustID IS '7173';
FOR ALL NEXT TransDate Title INTO CustMovies
    WHERE Category IS 'COMEDY';
```
The structure of the joined VideoTrk and Movies data sources looks like this:

In this NEXT command, the TransDat segment is the anchor and the MovInfo segment is the target. The resulting CustMovies stack contains all the fields needed to define the data source path from the root segment to the target segment:

- The ancestor segment of the anchor, Cust (key field only).
- All segments from the anchor through the root: TransDat, Rentals, MovInfo (all fields).

The stack looks like this:

<table>
<thead>
<tr>
<th>CustID</th>
<th>TransDate</th>
<th>MovieCode</th>
<th>...</th>
<th>Title</th>
<th>...</th>
<th>Copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>7173</td>
<td>91/06/18</td>
<td>305PAR</td>
<td>...</td>
<td>AIRPLANE</td>
<td>...</td>
<td>2</td>
</tr>
</tbody>
</table>
Creating a Data Source Stack User-defined Columns

In addition to creating data source stack columns that correspond to data source fields, you can also create data source stack columns that you define yourself. You can define these columns in two ways:

- **Within a procedure.** You can create a stack column (as well as user-defined variables) by issuing a COMPUTE command. You can also use the COMPUTE command to assign values to stack cells.

  Because all Maintain variables are local to a procedure, you must redefine variables in each procedure in which you use them. For user-defined stack columns, you accomplish this by simply reissuing the original COMPUTE command in each procedure to which you are passing the stack. You only need to specify the format of the variable. You do not need to specify its value, which is passed with the stack.

- **Within the Master File.** You can define a virtual field in a Master File by using the DEFINE attribute. The field is then available in every procedure that accesses the data source. The virtual field is treated as part of the data source segment in which it is defined, and Maintain automatically creates a corresponding column for it, a virtual column, in every stack that references that segment.

  Virtual fields must be derived, directly or indirectly, from data source values. They cannot be defined as a constant. The expression assigned to a virtual field in the Master File can reference fields from other segments in the same data source path as the virtual field.

  Unlike other kinds of stack columns, you cannot update a virtual column or field, and you cannot test it in a WHERE phrase.
Example: Creating a User-defined Column

Consider a data source stack named Pay that contains information from the Employee data source. If you want to create a user-defined column named Bonus and set its value to 10% of each employee’s current salary, you could issue the COMPUTE command to create the new column, and then issue another COMPUTE to derive the value. You place the second COMPUTE within a REPEAT loop to run it once for each row in the stack:

```plaintext
COMPUTE Pay.Bonus/D10.2;
REPEAT Pay.FocCount  Row/I4=1;
    COMPUTE Pay(Row).Bonus = Pay(Row).Curr_Sal * .10;
ENDREPEAT  Row=Row+1;
```

Copying Data Into and Out of a Data Source Stack

You can copy data into and out of a data source stack in the following ways:

- **Between a stack and a data source.** You can copy data from a data source into a stack using the NEXT and MATCH commands. You can copy data in the opposite direction, from a stack into a data source, using the INCLUDE, UPDATE, and REVISE commands. In addition, the DELETE command, while not actually copying a stack data, reads a stack to determine which records to remove from a data source. For more information about these commands, see Command Reference in the Maintain Language Reference manual.

- **Between a stack and a form.** You can copy data from a stack into a form, and from a form into a stack, by specifying the stack as the source or destination of the data displayed by the form. This technique makes it easy for an application user to enter and edit stack data at a personal computer. For more information about using stacks with forms, see Developing and Using Controls in the Developing WebFOCUS Maintain Applications manual.

- **From a transaction file to a stack.** You can copy data from a transaction file to a stack using the NEXT command. For more information about this command, see the Maintain Language Reference manual.

- **Between two stacks.** You can copy data from one stack to another using the COPY and COMPUTE commands. For more information about these commands, see Command Reference in the Maintain Language Reference manual.

You can use any of these commands to copy data by employing the command INTO and FROM phrases. FROM specifies the command data source (the source stack), and INTO specifies the command data destination (the destination stack).
Example: Copying Data Between a Data Source Stack and a Data Source

In this NEXT command

\[
\text{FOR ALL NEXT CustID INTO CustStack;}
\]

the INTO phrase copies the data (the CustID field and all of the other fields in that segment) into CustStack. The following UPDATE command

\[
\text{FOR ALL UPDATE ExpDate FROM CustStack;}
\]

uses the data from CustStack to update records in the data source.

Referring to Specific Stack Rows Using an Index

Each stack has an index that enables you to refer to specific rows. For example, by issuing a NEXT command, you create the CustNames stack to retrieve records from the VideoTrk data source:

\[
\text{FOR ALL NEXT CustID LastName INTO CustNames WHERE ExpDate GT 920624;}
\]

The first record retrieved from VideoTrk becomes the first row in the data source stack, the second record becomes the second row, and so on.

<table>
<thead>
<tr>
<th>CustID</th>
<th>LastName</th>
<th>...</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0925</td>
<td></td>
<td>61601</td>
</tr>
<tr>
<td>2</td>
<td>1118</td>
<td></td>
<td>61601</td>
</tr>
<tr>
<td>3</td>
<td>1423</td>
<td></td>
<td>61601</td>
</tr>
<tr>
<td>4</td>
<td>2282</td>
<td></td>
<td>61601</td>
</tr>
<tr>
<td>5</td>
<td>4862</td>
<td></td>
<td>61601</td>
</tr>
<tr>
<td>6</td>
<td>8771</td>
<td></td>
<td>61601</td>
</tr>
<tr>
<td>7</td>
<td>8783</td>
<td></td>
<td>61601</td>
</tr>
<tr>
<td>8</td>
<td>9022</td>
<td></td>
<td>61601</td>
</tr>
</tbody>
</table>
You can refer to a row in the stack by using a subscript. The following example refers to the third row, in which CustID is 1423:

```
CustNames(3)
```

You can use any integer value as a subscript: an integer literal (such as 3), an integer field (such as TransCode), or an expression that resolves to an integer (such as TransCode + 2).

You can even refer to a specific column in a row (that is, to a specific stack cell) by using the stack name as a qualifier:

```
CustNames(3).LastName
```

If you omit the row subscript, the position defaults to the first row. For example,

```
CustNames.LastName
```

is equivalent to

```
CustNames(1).LastName
```

Maintain provides two system variables associated with each stack. These variables help you to navigate through a stack and to manipulate single rows and ranges of rows:

- **FocCount.** This value of the variable is always the number of rows currently in the stack and is set automatically by Maintain. This is very helpful when you loop through a stack, as described in the following section, *Looping Through a Stack* on page 153. FocCount is also helpful for checking if a stack is empty:

  ```
  IF CustNames.FocCount EQ 0 THEN PERFORM NoData;
  ```

- **FocIndex.** This variable points to the current row of the stack. When a stack is displayed in a form, the form sets FocIndex to the row currently selected in the grid or browser. Outside of a form, the developer sets the value of FocIndex. By changing its value, you can point to any row you wish. For example, in one function you can increment FocIndex for the Rental stack:

  ```
  IF Rental.FocIndex LT Rental.FocCount
  THEN COMPUTE Rental.FocIndex = Rental.FocIndex + 1;
  ```

You can then invoke a second function that uses FocIndex to retrieve desired records into the MovieList stack:

```
FOR ALL NEXT CustID MovieCode INTO MovieList
  WHERE VideoTrk.CustID EQ Rental(Rental.FocIndex).CustID;
```

The syntax "stackname(stackname.FocIndex)" is identical to "stackname()", so you can code the previous WHERE phrase more simply as follows:

```
WHERE VideoTrk.CustID EQ Rental().CustID
```
Looping Through a Stack

The `REPEAT` command enables you to loop through a stack. You can control the process in different ways, so that you can loop according to several factors:

- The number of times specified by a literal, or by the value of a field or expression.
- The number of rows in a stack, by specifying the FocCount variable of the stack.
- While an expression is true.
- Until an expression is true.
- Until the logic within the loop determines that the loop should be exited.

You can also increment counters as part of the loop.

Example: Using REPEAT to Loop Through a Stack

The following `REPEAT` command loops through the Pay stack once for each row in the stack and increments the temporary variable Row by one for each loop:

```plaintext
REPEAT Pay.FocCount  Row/I4=1;
    COMPUTE Pay(Row).NewSal = Pay(Row).Curr_Sal * 1.10;
ENDREPEAT  Row=Row+1;
```

Sorting a Stack

You can sort the row of a stack using the `STACK SORT` command. You can sort the stack by one or more of its columns and sort each column in ascending or descending order. For example, the following `STACK SORT` command sorts the CustNames stack by the LastName column in ascending order (the default order):

```plaintext
STACK SORT CustNames BY LastName
```

Editing and Viewing Stack Values

There are multiple ways in which you can edit and/or view the values of a stack.

- **Forms.** You can display a stack in an HTML table or a grid on a form. A grid enables you to edit the fields of a stack directly on the screen. You cannot edit a stack in an HTML table.
COMPUTE command. You can use the COMPUTE command to assign a value to any of the cells of a stack. When assigning a value, the COMPUTE keyword is optional, as described in Command Reference in the Maintain Language Reference manual. For example, the following command assigns the value 35000 to the cell at the intersection of row 7 and column NewSal in the Pay stack:

```
COMPUTE Pay(7).NewSal = 35000;
```

It is important to note that if you do not specify a row when you assign values to a stack, Maintain defaults to the first row. Thus, if the Pay stack has 15 rows and you issue the following command

```
COMPUTE Pay.NewSal = 28000;
```

the first row receives the value 28000. If you issue this NEXT command

```
FOR 6 NEXT NewSal INTO Pay;
```

the current row of Pay defaults to one, and so the six new values are written to rows one through six of Pay. Any values originally in the first six rows of Pay will be overwritten.

If you wish to append the new values to Pay, that is, to add them as new rows 16 through 21, you would issue this NEXT command, which specifies the starting row:

```
FOR 6 NEXT NewSal INTO Pay(16);
```

You can accomplish the same thing without needing to know the number of the last row by of the stack using the FocCount variable:

```
FOR 6 NEXT NewSal INTO Pay(Pay.FocCount+1);
```

If you want to discard the original contents of Pay and substitute the new values, it is best to clear the stack before writing to it using the following command:

```
STACK CLEAR Pay;
FOR 6 NEXT NewSal INTO Pay;
```
Default Data Source Stack: The Current Area

For all data source fields referenced by a Maintain procedure, Maintain creates a corresponding column in the default data source stack known as the **Current Area**.

The Current Area is always present and is global to the procedure. It has one row, and functions as a kind of data source buffer. Each data source field, that is, each field described in a Master File that is accessed by a Maintain procedure, has a corresponding column in the Current Area. When a data source command assigns a value, either to a field using INCLUDE, UPDATE, or REVISE, or from a field to a stack using NEXT or MATCH, Maintain automatically assigns that same value to the corresponding column in the single row of the Current Area. If a set-based data source command writes multiple values to or from a stack column, the last value that the command writes is the one that is retained in the Current Area.

**Note:** Stacks are a superior way of manipulating data source values. Since the Current Area is a buffer, it does not function as intuitively as stacks do. It is recommended that you use stacks instead of the Current Area to manipulate data source values.

For example, if you write 15 values of NewSal to the Pay stack, the values will also be written to the NewSal column in the Current Area; since the Current Area has only one row, its value will be the fifteenth (that is, the last) value written to the Pay stack.

The Current Area is the default stack for all FROM and INTO phrases in Maintain commands. If you do not specify a FROM stack, the values come from the single row in the Current Area. If you do not specify an INTO stack, the values are written to the single row of the Current Area, so that only the last value written remains.

The standard way of referring to a stack column is by qualifying it with the stack name and a period:

```
stackname.columnname
```

Because the Current Area is the default stack, you can explicitly reference its columns without the stack name, by prefixing the column name with a period:

```
.columnname
```

Within the context of a WHERE phrase, an unqualified name refers to a data source field (in a NEXT command) or a stack column (in a COPY command). To refer to a Current Area column in a WHERE phrase you should reference it explicitly by qualifying it with a period. Outside of a WHERE phrase it is not necessary to prefix the name of a Current Area column with a period, as unqualified field names will default to the corresponding column in the Current Area.
For example, the following NEXT command compares Emp_ID values taken from the Employee data source with the Emp_ID value in the Current Area:

```
FOR ALL NEXT Emp_ID Pay_Date Ded_Code INTO PayStack
    WHERE Employee.Emp_ID EQ .Emp_ID;
```

If the Current Area contains columns for fields with the same field name but located in different segments or data sources, you can distinguish between the columns by qualifying each one with the name of the Master File and/or segment in which the field is located:

```
masterfile_name.segment_name.column_name
```

If a user-defined variable and a data source field have the same name, you can qualify the name of the Current Area column of the data source field with its Master File and/or segment name; an unqualified reference will refer to the user-defined variable.

### Maximizing Data Source Stack Performance

When you use data source stacks, there are several things you can do to optimize performance:

- **Filter out unnecessary rows.** When you read records into a stack, you can prevent the stack from growing unnecessarily large by using the WHERE phrase to filter out unneeded rows.

- **Clear stacks when done with data.** Maintain automatically releases a stack memory at the end of a procedure, but if in the middle of a procedure you no longer need the data stored in a stack, you can clear it immediately by issuing the STACK CLEAR command. Clearing the data frees the stack memory for use elsewhere.

- **Do not reuse a stack for an unrelated purpose.** When you specify a stack as a data source or destination in certain contexts (in the NEXT, MATCH, and INFER commands, and in the Form Editor for controls), you define the columns that the stack will contain. If you use the same stack for two unrelated purposes, it will be created with the columns needed for both, making it unnecessarily wide.

### Controlling the Flow of a Procedure

Maintain provides many different ways of controlling the flow of execution within a procedure. You can:

- Nest a block of code. In commands in which you can nest another command, such as in an IF command, you can nest an entire block of commands in place of a single one by defining the block using the BEGIN command.

- Loop through a block of code a set number of times, while a condition remains true, or until it becomes true, using the REPEAT command.
Branch unconditionally to a block of code called a Maintain function. You define the function using the CASE command, and can invoke it in a variety of ways. When the function terminates, it returns control to the command following function invocation.

Alternatively, you can branch to a function, but not return upon termination, by invoking the function using the GOTO command.

Branch conditionally using the IF command. If the expression you specify in the IF command is true, the command executes a PERFORM or GOTO command nested in the THEN phrase, which branches to a Maintain function.

Alternatively, you can nest a different command, such as a BEGIN command defining a block of code, to be conditionally run by the IF command.

Trigger an event handler in response to user events. When users perform the event in a form at run time, the event triggers the event handler, a function or URL link, that you have specified. Event handlers are described in *Forms and Event-driven Processing* on page 164, and in *Defining Events and Event Handlers* in the *Developing WebFOCUS Maintain Applications* manual.

For more information on the commands listed here, see *Command Reference* in the *Maintain Language Reference* manual.

**Executing Other Maintain Procedures**

You can call one Maintain procedure from another with the CALL command. *Maintain procedure* here means any procedure of Maintain language commands. CALL simplifies the process of modularizing an application. Software designed as a group of related modules tends to be easier to debug, easier to maintain, and lends itself to being reused by other applications, all of which increase your productivity.

CALL also makes it easy to partition an application, deploying each type of logic on the platform on which it will run most effectively.

The following diagram illustrates how to describe the relationship between called and calling procedures. It describes a sequence of five procedures from the perspective of the middle procedure, which is named C.
**Note:** A root procedure is also called the starting procedure.

---

**Calling a Maintain Procedure on a Different Server**

If parent and child Maintain procedures reside on different servers, you identify the location of the child procedure in a deployment scenario in the Maintain Development Environment. At deployment time, the Maintain Development Environment automatically informs the parent procedure of the location of the child procedure by supplying the AT server phrase in the CALL command. Do not code the AT server phrase yourself unless the parent procedure is not part of your project.

If the parent procedure is not a native project component, code the AT server phrase yourself in the CALL command of the parent procedure (or use the Language Wizard to generate the CALL command) to identify the location of the child procedure.
**Example:**  **Calling the EmpUpdat Procedure on a Different Server**

Consider the EmpUpdat procedure:

```equation
MAINTAIN FILE Employee 
FOR ALL NEXT Emp_ID INTO EmpStack;
.
.
.
CALL NewClass;
.
.
.
END
```

This calls the NewClass procedure that is deployed on the EducServ WebFOCUS Server:

```equation
MAINTAIN
.
.
.
END
```

In this example, EmpUpdat is the parent procedure and NewClass is the child procedure. When the child procedure, and any procedures that it has invoked, have finished executing, control returns to the parent.

**Passing Variables Between Procedures**

All user variables (both stacks and simple, or scalar, variables) are global to a function or procedure, but not global to the project. In other words, to protect them from unintended changes in other parts of a project, you cannot directly reference a variable outside of the procedure in which it is found (with the exception of the FocError transaction variable). However, you can access the variable data in other procedures, simply by passing it as an argument from one procedure to another.

To pass variables as arguments, you only need to name them in the CALL command, and then again in the corresponding MAINTAIN command, using the FROM phrase for input arguments and INTO phrase for output arguments. Some variable attributes must match in the CALL and MAINTAIN commands:

- **Number.** The number of arguments in the parent and child procedures must be identical.
- **Sequence.** The order in which you name stacks and simple variables must be identical in the CALL and corresponding MAINTAIN commands.
- **Data type.** Stack columns and simple variables must have the same data type (for example, integer) in both the parent and child procedures.
Stack column names. The names of stack columns need to match. If a column has different names in the parent and child procedures, it is not passed.

Other attributes need not match:

- **Stack and scalar variable names.** The names of stacks and simple variables specified in the two commands need not match.

- **Other data attributes.** All other data attributes, such as length and precision, do not need to match.

- **Simple variables.** If you pass an individual stack cell, you must receive it as a simple variable, not as a stack cell.

After you have passed a variable to a child procedure, you need to define it in that procedure. How you define it depends upon the type of variable:

- **User-defined columns and fields.** You must redefine each user-defined variable using a DECLARE or COMPUTE command. You only need to specify the variable format, not its value. For example, the following DECLARE command redefines the Counter field and the FullName column:

  DECLARE Counter/A20;
  EmpStack.FullName/A15;

- **Data source and virtual stack columns.** You can define the data source columns and virtual columns of the stack in one of two ways. You can define them implicitly, by referring to the stack columns in a data source command, or explicitly, by referring to them using the INFER command. For example:

  INFER Emp_ID Pay_Date INTO EmpStack;

  The INFER command declares data source fields and the stack with which they are associated. You can specify one field for each segment you want in the stack or simply one field each from the anchor and target segments of a path you want in the stack.

  While INFER reestablishes the definition of the stack, it does not retrieve any records from the data source.

After a variable has been defined in the child procedure, its data becomes available. If you refer to stack cells that were not assigned values in the parent procedure, they are assigned default values (such as spaces or zeros) in the child procedure, and a message is displayed warning that they have not been explicitly assigned any values.

When the child procedure returns control back to the parent procedure, the values of stacks and simple variables specified as output arguments are passed back to the parent. The values of stacks and simple variables specified only as input arguments are not passed back.
**Example:** Passing Data Between Maintain Procedures

This procedure

```maintain
MAINTAIN FILE Employee
FOR ALL NEXT Emp_ID INTO EmpStack;
.
.
CALL NewClass FROM EmpStack CourseStack INTO CourseStack;
.
.
END
```
calls the NEWCLASS procedure:

```maintain
MAINTAIN FROM StudentStack CourseStack INTO CourseStack
.
.
END
```

EmpStack and CourseStack in the parent procedure correspond to StudentStack and CourseStack in the child procedure.

**Accessing Data Sources in the Child Procedure**

If a child procedure accesses a data source, whether retrieving or writing records, you must specify the data source in the MAINTAIN command. This is done the same way as for a stand-alone procedure. For example, the procedure below specifies the Employee and EducFile data sources:

```maintain
MAINTAIN FILES Employee AND EducFile FROM StuStk INTO CoursStk
.
.
END
```

**Data Source Position in Child Procedures**

Each Maintain procedure tracks its own position in the data source. When you first call a procedure, Maintain positions you at the beginning of each segment in each data source accessed within that procedure. After navigating through a data source, you can reposition to the beginning of a segment by issuing the REPOSITION command. The data source positions are independent of the positions established in other procedures.
When a child procedure returns control to its parent, by default it clears its data source positions. You can specify that it retain its positions for future calls by using the KEEP option, as described in Optimizing Performance: Data Continuity and Memory Management on page 162.

**Advantages of Modularizing Source Code Using CALL**

Modularizing source code into several procedures has many advantages. One benefit is that you can use multiple procedures, run using the CALL command, to share common source code among many developers, speeding up both development and maintenance time. For example, a generalized error message display procedure could be used by all WebFOCUS Maintain developers. After passing a message to the generalized procedure, the procedure would handle message display. The developers do not need to worry about how to display the message, and the error messages will always look consistent to end users.

Another advantage of modular design is that you can remove infrequently-run source code from a procedure and move it into its own procedure. This reduces the size of the original procedure, simplifying its logic, making maintenance easier, and using less memory if the new procedure is not called.

**Optimizing Performance: Data Continuity and Memory Management**

By default, when you terminate a child procedure, Maintain clears its data from memory to save space. You can optimize your application performance by specifying, each time you terminate a child procedure, how you want Maintain to handle the procedure data. You have two options, based on how often you will call a given procedure over the course of an application. If you will call the procedure:

- Frequently, use the KEEP option to make the procedure run faster by retaining its data between calls.

  This option provides data continuity. The procedure data carries over from the end of one invocation to the beginning of the next. The next time you call the procedure, its variables and data source position pointers start out with the same values that they held when the procedure was last terminated. You can use these values or reinitialize them using the DECLARE (or COMPUTE) and REPOSITION commands.

  Of course, variables passed by the parent procedure are not affected by data continuity since the child procedure receives them directly from the parent procedure at the beginning of each call.
KEEP affects transaction integrity in the following way. The KEEP option does not issue an implied COMMIT command at the end of a child procedure. When a child procedure with an open logical transaction returns to its parent procedure and specifies KEEP, the transaction continues into the parent.

Rarely, use the RESET option to reduce memory consumption by freeing the procedure data at the end of each call.

This option does not provide data continuity; all of the procedure variables and data source position pointers are automatically initialized at the beginning of each procedure.

RESET affects transaction integrity in the following way. The RESET option issues an implied COMMIT command at the end of a child procedure. When a child procedure with an open logical transaction returns to its parent procedure using RESET, the transaction is closed at the end of the child procedure.

For more information about transactions spanning procedures, see Ensuring Transaction Integrity in the Developing WebFOCUS Maintain Applications manual.

You can specify how a procedure will handle its data in memory by terminating it with the GOTO END command qualified with the appropriate memory-management phrase. The syntax is

```
GOTO END [KEEP|RESET];
```

where:

**KEEP**

Terminates the procedure, but keeps its data, the values of its variables and data source position pointers, in memory. It remains in memory through the next invocation, or (if it is not called again) until the application terminates. The procedure does not issue an implied COMMIT command to close an open logical transaction.

**RESET**

Terminates the procedure, clears its data from memory, and issues an implied COMMIT command to close an open logical transaction. RESET is the default value.

You can use both options in the same procedure. For example, when you are ready to end a child procedure, you could evaluate what logic the procedure will need to perform when it is next called and then branch accordingly either to keep data in memory, saving time and providing data continuity, or else to clear data from memory to conserve space.
Forms and Event-driven Processing

Forms are the visual interface to a WebFOCUS Maintain application, giving it a dynamic and attractive face while enabling you to make the application flexible and to place its power at the fingertips of the application end users.

WebFOCUS Maintain supports a full set of controls, such as list boxes, buttons, and ActiveX components, by which end users can manipulate data and drive the application. You can design forms that enable end users to:

- Enter and edit data.
- Select options.
- Perform business logic, such as searching a data source for a customer order.
- Send email.
- Read application-specific help information.
- Control the flow of an application using an event-driven paradigm.

You develop these forms and the associated logic using the Form Editor. This is a sample form:

Forms are event-driven, and enable:

- **Event-driven processing.** Forms are responsive to the needs of users because they recognize user activity on the screen, that is, different types of screen events. For example, a form recognizes what the user does on the screen with the keyboard and mouse. It knows when users click a button or change a field value.
Forms also enable you to assign event handlers to these events. Each time a specified event occurs, Maintain automatically triggers the corresponding event handler. If you use events to trigger the application's business logic, you can give the user more freedom, for example, over which editing tasks to perform, and in which order. You can also give the user access to more functionality, and more types of data, on a single screen. Event-driven processing gives the user more flexibility over the application, even as it gives the application more control over the user interface.

- **Event-driven development.** WebFOCUS Maintain provides you with a simple way of developing event-driven applications, event-driven development. Because much of an application flow can be controlled from forms, you can develop the application as you paint its forms. You can first design the visual layout, then create controls, and finally code event handlers for the controls, all from the Form Editor. WebFOCUS Maintain also offers the Language Wizard which generates code for you, making it faster and easier to develop effective interfaces and powerful applications.

For an introduction to using forms and developing event-driven applications, see *How to Use Forms* on page 165, *Designing Event-driven Applications* on page 166, and *Creating Event-driven Applications* on page 167. Forms, and the Form Editor used to design and create them, are described in greater detail in *Developing and Using Forms* in the *Developing WebFOCUS Maintain Applications* manual.

### How to Use Forms

Forms are deployment-independent. You design a form to meet the needs of your application. WebFOCUS Maintain automatically implements the form as a webpage. This enables you to focus on logic, and leave implementation details to WebFOCUS Maintain.

Forms have standard form features, including:

- **A title bar that identifies the form.**

- **Scroll bars that enable you to move the contents of a control vertically and horizontally if they extend beyond the control border.**

Forms are displayed one at a time in one web browser session.

You can transfer control from one form to another, from a form to another Maintain procedure or to a WebFOCUS procedure, and from a form to any Internet resource, such as an email client, a webpage, or an FTP server.
Designing a Form

Forms offer a diverse set of ways by which an application end user can select options, invoke procedures, display and edit fields, and get helpful information. For example, if you want the user to select an option or procedure, you can use any of the following controls:

- **Buttons.** You can specify a function to be performed when the end user clicks a button or image. Common examples are Done and Cancel buttons.

- **Radio buttons.** The end user can select one of a mutually exclusive group of options. For example, an employee could identify his or her department.

- **Check boxes.** The end user can select or deselect a single option or characteristic. For example, an applicant could indicate if this is the first time that he or she has applied.

- **Combo boxes and list boxes.** The end user can select one or more options from a dynamic list of choices.

- **Menus.** The end user can select an option from a drop-down menu or submenu.

If you want to display or edit data, you can use these controls:

- **Edit boxes.** You can use this control to edit a single value.

- **Multi-line edit boxes.** You can use this control to edit a long value wrapping onto multiple lines.

- **HTML tables.** You can use this control to view a data source stack.

- **Grids.** You can use this control to edit and view a data source stack.

Designing Event-driven Applications

The flow of control in conventional processing is mostly pre-determined, that is, the programmer determines the few paths that the user will be able to take through the procedure.

To make your application user interface more responsive to the user, WebFOCUS Maintain offers event-driven processing. Each time that an event occurs, it invokes, or triggers, the assigned event handler. In WebFOCUS Maintain, the event is something the application end user does in a form, and the event handler is a function or a URL. For example, you might create a button that, when clicked by a user, triggers an event handler that reads a data source and displays the data in the form.
Creating Event-driven Applications

Developing a procedure by writing out sequential lines of source code may be sufficient for conventional linear processing, but event-driven processing demands event-driven development. Developing an application in this way lets you build much of the application logic around the user interface. In effect, you develop the application as you develop the interface in the Form Editor. For example, you could start by creating a form, creating a control, and then coding an event handler for one of the control events. WebFOCUS Maintain also provides you with a number of automated tools for developing applications. For example, you can use the Language Wizard to generate source code for various operations such as retrieving and updating data.

Reading From a Data Source

Most applications need to read data from a data source. The most common method is to read data from a data source into a data source stack. Before reading, you first need to select the record in which the data resides. There are five ways of selecting records:

- **By field value for an entire set of records.** Use the NEXT command. The WHERE phrase enables you to select by value, and the FOR ALL phrase selects the entire set of records that satisfy the WHERE selection condition. The basic syntax for this is:

  ```
  FOR ALL NEXT fields INTO stack WHERE selection_condition;
  ```

- **By field value for a sequence (subset) of records.** Use the NEXT command. This is similar to the technique for a set of records, except that it employs the FOR n phrase, selecting, at the current position in the data source, the first n records that satisfy the WHERE condition. The basic syntax for this is:

  ```
  FOR n NEXT fields INTO stack WHERE selection_condition;
  ```

- **By field value, one segment at a time, one record at a time.** Use the MATCH command. The basic syntax for this is:

  ```
  MATCH fields [FROM stack] [INTO stack];
  ```

- **Sequentially for a sequence (subset) of records.** Use the NEXT command. This technique employs the FOR n phrase to select the next n records. The basic syntax for this is:

  ```
  FOR n NEXT fields INTO stack;
  ```

- **Sequentially, one segment instance at a time, one record at a time.** Use the NEXT command. The basic syntax for this is:

  ```
  NEXT fields [INTO stack];
  ```
You can read from individual data sources, and from those that have been joined. Maintain supports joins that are defined in a Master File. For information about defining joins in a Master File, see the *Describing Data With WebFOCUS Language* manual. Maintain can read from joined data sources, but cannot write to them.

You can evaluate the success of a command that reads from a data source by testing the FocError system variable, as described in *Evaluating the Success of a Simple Data Source Command* on page 169.

The NEXT and MATCH commands are described in detail in *Command Reference* in the *Maintain Language Reference* manual.

**Repositioning Your Location in a Data Source**

Each time you issue a NEXT command, Maintain begins searching for records from the current position in the data source. For example, if your first data source operation retrieved a set of records

```plaintext
FOR ALL NEXT CustID INTO SmokeStack
    WHERE ProdName EQ 'VCR DUST COVER';
```

then Maintain will have searched sequentially through the entire data source, so the current position marker will now point to the end of the data source. If you then issue another NEXT command

```plaintext
FOR ALL NEXT LastName FirstName INTO CandyStack
    WHERE ProdName EQ 'CANDY';
```

Maintain searches from the current position to the end of the data source. Since the current position is the end of the data source, no records are retrieved.

When you want a NEXT command to search through the entire data source (as is often the case when you wish to retrieve a set of records) you should first issue the REPOSITION command to move the current position marker to the beginning of the data source.

**Example: Repositioning to the Beginning of the Data Source**

The following REPOSITION command specifies the CustID field in the root segment, and so moves the current position marker for the root segment chain and all of its descendant chains back to the beginning of the chain (in effect, back to the beginning of the data source):

```plaintext
REPOSITION CustID;
FOR ALL NEXT LastName FirstName INTO CandyStack
    WHERE ProdName EQ 'CANDY';
```
Writing to a Data Source

Writing to a data source is the heart of transaction processing applications. Maintain provides the following commands to write to a data source:

- **INCLUDE**: Adds the specified new segment instances to a data source.
- **UPDATE**: Updates the specified fields in a data source.
- **REVISE**: Adds new segment instances and updates the specified fields in existing segment instances.
- **DELETE**: Removes the specified segment instances from a data source.

These commands are described in detail in *Command Reference* in the *Maintain Language Reference* manual.

Maintain requires that data sources to which it writes have unique keys.

Evaluating the Success of a Simple Data Source Command

When you issue a command that reads or writes to a data source, you should determine if the command was successful. Reasons for a data source command not being successful include attempting to insert a record that already exists, to update a record that does not exist, to delete a record that does not exist, to read a record that does not exist, and being interrupted by a system failure.

When you issue a command that reads or writes to a data source, if the command is:

- Successful, Maintain automatically sets the transaction variable FocError to 0 (zero), and writes to the data source.
- Unsuccessful, Maintain sets FocError to a non-zero value, and does not write to the data source.

**Example:** Evaluating the Success of an UPDATE Command

The following function updates the VideoTrk data source for new video rentals. If the UPDATE command is unsuccessful, the application invokes a function that displays a message to the user. The line that evaluates the success of the command is highlighted below:

```maintain
CASE UpdateCustOrder
    UPDATE ReturnDate Fee FROM RentalStack;
    IF FocError NE 0 THEN PERFORM ErrorMessage;
ENDCASE
```
Evaluating the Success of a Stack-based Write Command

When you write a set of stack rows to a data source, if you specify more rows than there are matching data source records, this does not invalidate the write operation. Maintain attempts to write all the matching rows to the data source. For example, the following UPDATE command specifies 15 rows, but there are only 12 matching rows. All 12 are written to the data source.

FOR 15 UPDATE Curr_Sal FROM NewSalaries;

When you write a set of stack rows to a data source, if one row fails, the following happens:

- The rows preceding the failed row are written to the data source.
- The rows following the failed row are ignored.
- FocError is set to a non-zero value, signaling an error.
- FocErrorRow is set to the number of the failed row.

Data source logic errors include attempting to insert an existing record, to update a nonexistent record, and to delete a nonexistent record.

To determine if an entire stack was successfully written to the data source, test FocError immediately following the data source command. If FocError is not 0, you can determine which row caused the problem by testing FocErrorRow; you can then reprocess that row. If you will be passing control to a different procedure and reprocessing the row there, consider first setting the stack’s FocIndex variable to the value of FocErrorRow in the current procedure, so that after you pass control the stack is already positioned at the problem row.

If you do not wish to take advantage of this flexibility, and instead prefer to invalidate all the rows of the stack if any of them are unsuccessful, you can bracket the data source command in a logical transaction that you can then roll back. Logical transactions are discussed in Transaction Processing on page 171.

Row failure when committing to a data source. If a stack-based write command is part of a logical transaction, and the write command succeeds when it is issued but fails when the application tries to commit the transaction, Maintain rolls back all of the write command rows, along with the rest of the transaction. For example, a write command might fail at commit time because another user has already changed one of the records to which the command is writing. Transaction processing is described in Transaction Processing on page 171.
Example: Evaluating a Stack-based Update Command

The NewSalaries stack has 45 rows. The following command updates the Employee data source for all the rows in NewSalaries:

FOR ALL UPDATE Curr_Sal FROM NewSalaries;

If there is no data source record that matches the thirtieth row of NewSalaries, Maintain updates the data source records matching the first 29 rows and ignores records that match rows 30 and higher.

Transaction Processing

You are familiar with individual data source operations that insert, update, or delete data source segment instances. However, most applications are concerned with transactions, such as transferring funds or fulfilling a sales order, that each require several data source operations. These data source operations may access several data sources, and may be issued from several procedures. Such a collection of data source operations is called a logical transaction (and is also known as a logical unit of work.)

The advantage of describing a group of related data source commands as one logical transaction is that the transaction is written to the data source only if all of its component commands are successfully written to the data source. Transaction integrity is an uncompromising proposition: if even one part of the transaction fails when you try to write it (by issuing the COMMIT command), Maintain automatically rolls back the entire transaction, leaving the data source unchanged.

Transaction integrity also ensures that when several users share access to the same data source, concurrent transactions run as if they were isolated from each other. The changes caused by a transaction in a data source are concealed from all other transactions until that transaction is committed. This prevents each transaction from being exposed to interim inconsistent images of the data source, and so protects the data from corruption.

There are many strategies for managing concurrent data source access. No matter which type of data source you use, Maintain respects the DBMS concurrency strategy and lets it coordinate access to its own data sources.

Transaction processing is described in greater detail in Ensuring Transaction Integrity in the Developing WebFOCUS Maintain Applications manual.
**Example:** Logical Transactions in a Bank

A banking application would define a transfer of funds from a checking account to a savings account as one logical transaction comprising two update operations:

- Subtracting the funds from the source account (UPDATE Checking FROM SourceAccts).
- Adding the funds to the target account (UPDATE Savings FROM TargetAccts).

If the application had not been able to subtract the funds from the checking account, because someone had cleared a check against that account a few moments earlier and depleted its funds, but the application had added the funds to the savings account, the bank accounts would become unbalanced.

The two update commands (subtracting and adding funds) must be described as parts of a single logical transaction, so that the subtraction and addition updates are not written to the data source independently of each other.

### Classes and Objects

Most application development is modular, the developer creates complex systems comprised of smaller parts. In conventional development, these modules are processes (such as procedures). In object-oriented development, the modules are models of real-world objects (such as a customer or a shipping order). Each object encapsulates both data and processes.

For example, if you are developing an order fulfillment system for a mail-order clothing business, the objects might include customers, orders, and stock items. The customer data might include the ID code, phone number, and order history. the customer processes might include a function that adds the customer to a new mailing list, a function that updates the customer contact information, and a function that places an order for the customer.

Object-oriented development, because it models the real-world objects with which your enterprise deals, and encourages you to reuse application logic in a variety of ways, is a more efficient way of developing applications. WebFOCUS Maintain enables you to create applications using object-oriented development, conventional development, or a hybrid of these two methods, providing you with a flexible path.

### What Are Classes and Objects?

Most applications need many objects of the same type. For example, if your business has 500 customers, you need one object to represent each customer. No one would want to design a customer object 500 times. Clearly, you need a template that defines all customer objects, so that you can design the template once, and use it often, each time you create a new customer object to represent a new customer.
The template of an object is called its class. Each object is an instance of a class. The class defines what type of object it is. When you create a class, it becomes a new data type, one which you can use to define an object, in the same way that you can use a built-in data type like integer or alphanumeric to define a simple variable like a customer code.

You define a class by describing its properties. Classes have two kinds of properties:

- **Data**, in the form of the class variables. Because these variables exist only as members of the class, they are called *member variables*. In some object-oriented development environments, these are also known as object attributes or instance variables.

- **Processes**, implemented as functions. Because these functions exist only as members of the class, they are called *member functions*. In some object-oriented development environments, these are also known as methods.

If you want to create a new class that is a special case of an existing class, you could derive it from that existing class. For example, in a human resources application, a class called Manager could be considered a special case of a more general class called Employee. All managers are employees, and possess all employee attributes, plus some additional attributes unique to managers. The Manager class is derived from the Employee class, so Manager is a subclass of Employee, and Employee is the superclass of Manager.

A subclass inherits all of its superclass properties, that is, it inherits all of the superclass member variables and member functions. When you define a subclass you can choose to override some of the inherited member functions, meaning that you can recode them to suit the ways in which the subclass differs from the superclass. You can also add new member functions and member variables that are unique to the subclass.
Sample data sources have been used in examples throughout these manuals in order to provide meaningful examples.

For information on the standard sample data sources, see the *Describing Data With WebFOCUS Language* manual.

This chapter contains the Master Files and structure diagrams of the Fannames, Users, and Contact data sources, which are used exclusively in the WebFOCUS Maintain manuals.

You can find these sample files in the approot/maintain directory.

**In this appendix:**
- Fannames Data Source
- Users Data Source
- Contact Data Source

**Fannames Data Source**

The Fannames data source contains email, address, and telephone information for all fans in a fanclub.

**Fannames Master File**

```plaintext
FILENAME=FANNAMES, SUFFIX=FOC
SENGNAME=CUSTOMER, SEGTYPE=S1
FIELD=SSN, ALIAS=SSN, FORMAT=A11, $  
FIELD=LASTNAME, ALIAS=LASTNAME, FORMAT=A10, $  
FIELD=FIRSTNAME, ALIAS=FIRSTNAME, FORMAT=A8, $  
FIELD=COMPANY, ALIAS=COMPANY, FORMAT=A12, $  
FIELD=ADDRESS, ALIAS=ADDRESS, FORMAT=A20, $  
FIELD=CITY, ALIAS=CITY, FORMAT=A10, $  
FIELD=STATE, ALIAS=STATE, FORMAT=A2, $  
FIELD=ZIP, ALIAS=ZIP, FORMAT=A5, $  
FIELD=PHONE, ALIAS=PHONE, FORMAT=A15, $  
FIELD=EMAIL, ALIAS=EMAIL, FORMAT=A20, $  
FIELD=TITLE, ALIAS=TITLE, FORMAT=A4, $  
FIELD=USER, ALIAS=USER, FORMAT=A9, $  
```
The Users data source contains personal information about the types of users in a fanclub.

**Users Master File**

FILENAME=FANNAMES, SUFFIX=FOC  
SEGNAME=CUSTOMER, SEGTYPE=S1

FIELD=USER, ALIAS=USER, FORMAT=A9, $  
FIELD=PASS, ALIAS=PASS, FORMAT=A9, $  
FIELD=GROUP, ALIAS=GROUP, FORMAT=A15, $  

**Contact Data Source**

The Contact data source contains name, address, telephone, title, and position for all contacts.
Contact Master File

FILENAME=CONTACT, SUFFIX=FOC
SEGNAME=CUSTOMER, SEGTYPE=S1

FIELD=LAST, ALIAS=LAST, FORMAT=A10, $
FIELD=FIRST, ALIAS=FIRST, FORMAT=A8, $
FIELD=COMPANY, ALIAS=COMPANY, FORMAT=A12, $
FIELD=ADDRESS, ALIAS=ADDRESS, FORMAT=A20, $
FIELD=CITY, ALIAS=CITY, FORMAT=A10, $
FIELD=STATE, ALIAS=STATE, FORMAT=A2, $
FIELD=ZIP, ALIAS=ZIP, FORMAT=A5, $
FIELD=PHONE, ALIAS=PHONE, FORMAT=A15, $
FIELD=TITLE, ALIAS=TITLE, FORMAT=A6, $
FIELD=POSITION, ALIAS=POSITION, FORMAT=A15, $

Contact Structure Diagram

CUSTOMER S1

LAST
FIRST
COMPANY
<table>
<thead>
<tr>
<th><strong>Glossary</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access file</strong></td>
<td>Contains information needed by WebFOCUS to access the data in a data source. The Access File includes the real name and location of the data source.</td>
</tr>
<tr>
<td><strong>application</strong></td>
<td>A group of procedures and other components that work together to achieve a common goal. An application can be distributed across many servers, access varied data sources, and support multiple users.</td>
</tr>
<tr>
<td><strong>Boolean expression</strong></td>
<td>An expression in a programming language that returns a value of true (1) or false (0), based on the outcome of two or more relational expressions. Boolean expressions are often used in conditional expressions.</td>
</tr>
<tr>
<td><strong>class</strong></td>
<td>The template of an object. A class defines what type of object it is, and what it can do using variables and functions.</td>
</tr>
<tr>
<td><strong>Controls Palette</strong></td>
<td>A component in the Form Editor that contains the controls you can place on a Maintain form, such as text, radio buttons, check boxes and so on.</td>
</tr>
<tr>
<td><strong>data access logic</strong></td>
<td>The code that extracts data from the data source, manipulates it, and writes it back to the data source. All data access logic is coded using Maintain language.</td>
</tr>
<tr>
<td><strong>data source stack</strong></td>
<td>A non-persistent (or in-memory) table where you can store and manipulate data from one or more data sources. WebFOCUS Maintain procedures use stacks to hold values read from the data source and to manipulate data before writing it back to the data source.</td>
</tr>
<tr>
<td><strong>database logic</strong></td>
<td>The code that implements NEXT, INCLUDE, UPDATE, DELETE, and REVISE commands.</td>
</tr>
<tr>
<td><strong>DBA (Database Administrator)</strong></td>
<td>A user, whose role is to perform development, design, and monitoring of the database. A DBA has unlimited access to the Master File and data source, and can set up or change security restrictions for individual users.</td>
</tr>
<tr>
<td><strong>deep copy</strong></td>
<td>Copies both the appearance of a control and any underlying data bindings, event handlers, and connections to resources. See shallow copy.</td>
</tr>
<tr>
<td><strong>deployment scenario</strong></td>
<td>Maps all of the components in your project to a WebFOCUS Server or a web server.</td>
</tr>
<tr>
<td><strong>event</strong></td>
<td>An action that an end user performs, such as opening a form, clicking a button, or moving to a field.</td>
</tr>
<tr>
<td><strong>event handler</strong></td>
<td>A programmable response to an action performed by a user, such as entering text in a field or clicking a button.</td>
</tr>
<tr>
<td><strong>form</strong></td>
<td>An interface object that enables you to interact with an application. You can create a Maintain application containing multiple forms.</td>
</tr>
<tr>
<td><strong>Form Editor</strong></td>
<td>A tool in the Maintain Development Environment (MDE) that enables you to design the user interface for your Maintain application.</td>
</tr>
<tr>
<td><strong>front end</strong></td>
<td>The user interface with which the end user interacts. This generally includes forms, web links, and scripts.</td>
</tr>
<tr>
<td><strong>function</strong></td>
<td>A series of commands in a procedure grouped together as a unit of control. Also known as cases.</td>
</tr>
<tr>
<td><strong>import module</strong></td>
<td>A non-executable procedure that contains Maintain language code.</td>
</tr>
<tr>
<td><strong>Java applets</strong></td>
<td>Compiled Java programs that can perform interactive animation, immediate calculations, or other simple tasks without having to send a user request back to the server.</td>
</tr>
<tr>
<td><strong>JavaScript</strong></td>
<td>A small, object-based scripting language that you can use to create functions that can be embedded in or called from HTML pages. When you execute your application, your browser will interpret the code and perform the actions required.</td>
</tr>
<tr>
<td><strong>logical transaction</strong></td>
<td>A collection of data source operations that can access several data sources and that can be issued from several procedures.</td>
</tr>
<tr>
<td><strong>Master File</strong></td>
<td>A file that describes the data used by WebFOCUS Maintain. The extensions are .mas and .acx.</td>
</tr>
<tr>
<td><strong>member functions</strong></td>
<td>A class property that defines what you can do to objects of a class, and how you can manipulate the data. They exist only as members of the class.</td>
</tr>
<tr>
<td><strong>member variables</strong></td>
<td>A class property that defines data in the form of variables, existing only in the class.</td>
</tr>
<tr>
<td><strong>N-Tier Applications</strong></td>
<td>Applications capable of distributing processing over multiple platforms, which allows the creation of relationships among disparate data sources and increased application speed.</td>
</tr>
<tr>
<td><strong>numeric expression</strong></td>
<td>An expression in a programming language that performs a calculation using numeric constants, fields, operators, and functions.</td>
</tr>
<tr>
<td><strong>object</strong></td>
<td>An instance of a class that encapsulates both data and processes.</td>
</tr>
<tr>
<td><strong>Output window</strong></td>
<td>An area in the Maintain Development Environment (MDE) that displays messages generated by WebFOCUS Maintain as well as Search Project results.</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>persistence logic</strong></td>
<td>When an application processes a Winform_Show command, a formatted HTML screen is returned to the user, and the procedure and processing agent &quot;waits,&quot; or persists, until the user invokes another submit through the Servlet back to the running procedure. The Attach Manager of the server, using the cookie, reattaches the user to the waiting procedure and processing agent, and the processing continues. In this way, the waiting procedure and processing agent retain the context of the user between the different forms or screens of the application user experience.</td>
</tr>
<tr>
<td><strong>presentation logic</strong></td>
<td>Maintain syntax and browser script code that describes and implements the user interface for the application.</td>
</tr>
<tr>
<td><strong>pre-database stack processing</strong></td>
<td>Data source stacks that must be fully transformed to match the data control being used on the form before being returned to presentation logic procedure.</td>
</tr>
<tr>
<td><strong>property sheet</strong></td>
<td>Enables you to view and edit properties and events for your forms and controls.</td>
</tr>
<tr>
<td><strong>Resource Wizard</strong></td>
<td>A tool that helps you add resources, such as image files, HTML files, Java applets, and ActiveX controls to your project. Resources are non-native Maintain project components that you can use in the forms you design.</td>
</tr>
<tr>
<td><strong>script</strong></td>
<td>A component in the WebFOCUS Maintain project that contains stand-alone functions, written in either JavaScript or VBScript for execution on the client.</td>
</tr>
<tr>
<td><strong>set-based processing</strong></td>
<td>Enables you to read, manipulate, and write groups of records at a time. The data is manipulated using a data source stack.</td>
</tr>
<tr>
<td><strong>shallow copy</strong></td>
<td>Copies the appearance of the control. See deep copy.</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td><strong>starting procedure</strong></td>
<td>A procedure that you designate to be executed first, when running a project from the development environment.</td>
</tr>
<tr>
<td><strong>Synonym Editor</strong></td>
<td>A tool that enables you to view and edit the attributes of a synonym.</td>
</tr>
<tr>
<td><strong>transaction integrity</strong></td>
<td>A condition where the transaction is written to the data source only if all of its component commands are successfully written to the data source. If one part of the transaction fails, Maintain automatically rolls back the entire transaction, leaving the data source unchanged. Transaction integrity also ensures that when several users share access to the same data source, concurrent transactions run as if they were isolated from each other.</td>
</tr>
<tr>
<td><strong>user context</strong></td>
<td>The value of the variables and stacks of an application, the position in the database, and the state of the transaction at any given time.</td>
</tr>
<tr>
<td><strong>variable</strong></td>
<td>A named storage location capable of containing a certain type of data that can be modified during program execution.</td>
</tr>
<tr>
<td><strong>VBScript</strong></td>
<td>A scripting language similar to JavaScript, except that it is only supported in Microsoft Internet Explorer.</td>
</tr>
<tr>
<td><strong>web link</strong></td>
<td>An object that contains a valid URL, such as a web page, email address, or FTP server.</td>
</tr>
</tbody>
</table>
Index

$$.Declarations comment 66

3GL programs 16

A

Access Files 19
Add parameters option 27
addafan.gif 34, 97
advanced tutorial 99
aligning controls 52, 53
anchor controls 52, 53
applications 16, 17, 20, 21, 29, 32, 34, 56, 81, 82, 83, 162, 164, 166, 167
application logic 16
closing at run time 21, 81, 82
deploying 83
developing 17
event-driven 164, 167
event-driven processing 166
exiting at run time 21
partitioning 20, 162
running 32, 83
running locally 29, 56
stored procedures 16
arguments for Maintain procedures 26

B

Back button 81
background images 90, 95, 96
tiling 95
BackgroundImage property 90, 96
basic tutorial 33
Binding the Selection Result dialog box 61
button control 64

C

C/C++ programs 16
CALL command 26, 27, 157, 158, 159, 161, 162
example 161
modularizing code 162
passing variables 159
CASE command 66
child procedures 157, 161
accessing data sources in 161
CICS transactions 16
classes 172
Click event 72
ClickLink event 101, 102, 119
ClickRow parameter 101, 102, 119
COBOL programs 16
columns in data source stacks 45, 149, 150
comments 66
COMMIT command 161
COMPUTE command 101, 102, 119, 149
concurrent processing 171
CONTACT data source 176
cross-referenced Help 138, 140
Control Columns dialog box 76
controls 48, 49, 50, 52, 53, 106, 107
aligning 52, 53
copying 106, 107
grouping 49, 50
moving 48
selecting multiple 52, 53
Controls palette 21, 47
Create a Project dialog boxes 35
Create new project file dialog box 36
cross-referenced data sources 167

Maintain Getting Started 185
Current Area columns 155
currfan.gif 34, 97

data access logic 20, 25

data continuity 162
data source descriptions 19, 30, 40, 87
deploying 30, 87
data source stacks 45, 62, 66, 67, 71, 72, 75, 104, 112, 129, 130, 141, 144, 145, 147, 149, 150, 151, 153, 155, 156
clearing 71, 72
columns 145, 147
compared to SPA 141
copying data 150, 151
creating 45, 144, 145
creating for report output 129, 130
Current Area 155
data source-derived columns 145
deleting rows 112
displaying columns on forms 104
drag and drop operations 104
editing 153
extracting data from data sources 75
implied columns 62
looping through 153
optimizing performance 156
rows 151
sorting 153
user-defined columns 149
writing to data sources 66, 67
data sources 16, 25, 40, 41, 42, 44, 45, 64, 67, 74, 75, 76, 118, 145, 161, 167, 168, 169, 170, 171
accessing in child procedures 161
adding fields to forms 44, 45
adding to projects 40
concurrent transactions 171
displaying data from 76, 118
expanding in Project Explorer 42
extracting data 75
joined data sources 145
data sources (continued)
keys 169
logical transactions 171
position in 161, 168
reading 167, 169
sharing 171
specifying in a Maintain procedure 25, 41
updating records 67
writing to 64, 169, 170
Declarations comment 66
deep copy 106, 107
DEFINE attribute 149
DELETE command 169
Deploy and run scenario button 90
Deploy folder 30
Deploying Application dialog box 90
deploying projects 56
deployment scenarios 30, 83, 87, 90
creating 83
defining 87
Display all files in the project paths button 19, 39
drag-and-drop operations 44, 104
adding data source fields to a form 44
displaying stack columns on forms 104

E

Edit Event Handlers button/command 21
email links 136
embedded joins 145, 167
reading 167
END command 66
ENDCASE keyword 66
Enter a List Item dialog box 59
Event Handler editor 21, 72
event-driven processing 164, 166, 167
events 21, 72, 164
Exit button 81, 82
Explorer 18, 38, 95
    using folders 95
Exploring toolbar 90
external procedures 17, 27

Fan Club application
    creating a welcome screen 124
    displaying fans 118
fan.gif 34, 97
FanClub application 33, 34, 44, 54, 56, 64, 74, 83, 100, 124
    creating a welcome screen 124
    creating forms 44
    displaying fans 74
    removing fans 100
    requirements 34
    running 83
    running locally 56
    saving 54
    writing to a data source 64
FANNAMES data source 34, 44, 66, 76, 107, 110, 118, 175
    adding fields to forms 44
    deleting records 110
    displaying data from 76, 118
    updating records 107
    using in procedures 66
fanrpt.fex 99, 129
fields 42, 44, 45, 49, 52, 149, 155
    adding to forms 44, 45
    Current Area and 155
    renaming prompts 49, 52
flow of control 153, 156
    looping 153
FocCount variable 112, 151
FocErrorRow variable 170
FocIndex variable 101, 102, 119, 151
FOCUS code 16, 27
FOCUS reports 129, 133
folders in the Explorer 96
FOR keyword 144
Form Editor 21, 47, 97, 167
    displaying rulers 97
Form menu 21
forms 21, 63, 74, 79, 81, 128, 164, 165, 166
    closing at run time 81
    creating 74, 166
    event-driven processing 166
    events 164
    invoking procedures 166
    linking 79
    naming 63
    navigating between forms 128
    selecting options 166
front-end logic 20, 21
functions 21, 25, 26, 41, 64, 66, 72
    assigning to events 21, 72
    creating 25, 26, 64
    editing 66
    viewing source code 66
GOTO command 162
graphics 90
Grid control 118
grid in Form Editor 48

Help 138, 140
    context-sensitive 138
    Maintain language 138, 140
Help menu 138
hierarchical data sources 42
HTML files 27, 30, 87
    deploying 30, 87
HTML Object control 129, 133
HTML Table control 76, 81, 101, 102
drilling down 101, 102
   links 101, 102
   viewing scroll bars 81
HTMTABLE format for FOCUS reports 129

I
Image Source dialog box 90, 96
images 27, 30, 87, 90, 95, 97
   deploying 30, 87
   tiling 95
implied columns 62
IMS/TM transactions 16
INCLUDE command 67, 169
INFER command 66
input parameters for a Maintain procedure 26

J
Java applets 27
JavaScript 117
JavaScript functions 21, 27, 118
   confirming a deletion 118
   using in applications 21
JavaScript libraries 30, 87, 136
   associating with forms 136
   creating 136
   deploying 30, 87

K
KEEP keyword 162
key fields 42, 169
   requirements 169

L
Language Wizard 25, 26, 67, 71, 72, 75
   adding records to a data source 67
   clearing data source stacks 71, 72
   extracting data from data sources 75
Layout toolbar 21
List Source dialog box 59
Local Deploy 30, 83
localhost 30, 87
logic in applications 20
logical transactions 171
   concurrent transactions 171

M
mailto keyword in JavaScript 136
MAINTAIN command 66
Maintain Controls palette 47
Maintain language 16, 25
Maintain Language Wizard 25, 26, 67, 71, 72, 75
   adding records to a data source 67
   clearing data source stacks 71, 72
   extracting data from data sources 75
Maintain procedures 25, 26, 30, 36, 46, 55, 56, 66, 87, 156, 157, 158, 159, 161, 164
   accessing data sources 25, 161
   child 157
   components 46
   creating 36
   data source position 161
   deploying 30, 87
   event driven 164
   flow of control 156
   passing variables 159, 161
   remote 157, 158, 159, 161
   selecting starting procedure 55, 56
   setting input and output parameters 26
   viewing source code 66
Master Files 19, 30, 87
   deploying 30, 87
MATCH command 145, 167
  defining stack columns 145
memory management 162
modular source code 162

N

n-tier applications 16
New Function dialog box 64
New Virtual Folder dialog box 96
NEXT command 75, 145, 167
  defining stack columns 145
  set-based processing 167

O

objects 172
operating systems 15
output parameters for Maintain procedures 26
Overflow property 81

P

parameters 26
partitions 20
performance 162
  memory management 162
pictures 90
platforms 15
presentation logic 20
Project Explorer 41, 42, 46
  expanding data sources 42
projects 18, 20, 29, 34, 35, 54, 55, 56, 83
  creating 18, 34, 35
  deploying 83
  partitioning 20
  running 56
  running locally 29, 56
projects (continued)
  saving 54
    Starting Object 55, 56
prompts for fields 49, 52
property sheet 21, 47, 49, 50

R

Radio button control 58, 59
RDBMS stored procedures 16
ReadOnly property 104
records 67, 107, 110, 142
  adding 67
  deleting 110, 142
  selecting 142
  updating 107, 142
relational data sources 42
remote procedures 157
REPEAT command 153
reports 28, 129, 130, 133, 135
  executing output into a stack 129, 133
 Maintain applications and 129, 130
  multiple copies 135
  starting Maintain procedure 28
REPOSITION command 75, 161, 168
reserved words 104
RESET keyword 162
Resource Wizard 90, 129
return values for Maintain procedures 26
rulers 97

S

Save button/command 54
saving your work 54
Scratch Pad Area (SPA) 141
INDEX

script functions 21, 27
  using in applications 21
script libraries 30, 87
  deploying 87
scroll bars for HTML tables 81
segments 42
Select Segment Fields dialog box 45
SelectedItem property 61
servers 16, 83
set-based processing 141, 142, 144, 167
  methods 144
shallow copy 106, 107
shortcuts 44, 104
  adding data source fields to a form 44
  displaying stack columns on forms 104
Show All Files command 19, 39
SHOW keyword in Winform command 66
showall.gif 99, 133
source code for procedures 66
SPA (Scratch Pad Area) 141
spiralbg.gif 34, 90
stacks 45, 62, 66, 67, 71, 72, 75, 112, 129, 135, 141, 144, 145, 147, 149, 150, 151, 153, 155, 156
  clearing 71, 72, 135
  columns 145, 147
  compared to SPA 141
  copying data 150, 151
  creating 45, 144, 145
  Current Area 155
  data source-derived columns 145
  editing 153
  extracting data from data sources 75
  for report output 129
  implied columns 62
  looping through 153
  optimizing performance 156
  rows 112, 151
  segments 145
  user-defined columns 149
stacks (continued)
  writing to data sources 66, 67
Starting Object for projects 55, 56
starting procedure 55, 56
syntax color 66

T

Table Column dialog box 76, 78
temporary fields 149, 159
  redefining in child procedures 159
Text control 104
Text property 49, 50
Title property 63
Toggle grid button 48
Toggle rulers button 97
ToolTipText property 61
Top function 46, 66
transaction processing 171
  concurrent transactions 171
transaction values 142
triggers 164
tutorials 33, 99
Type Wizard 114

U

untitled forms 63
UPDATE command 169
Use these Data Sources in Procedure dialog box 41
user-defined stack columns 149, 150
USERS data source 176
V
variables 112, 114, 159, 161, 162
memory management 162
passing between procedures 159, 161
VBScript functions 21, 27
using in applications 21
VBScript libraries 30, 87
deploying 87
virtual fields 149
virtual folders in the Explorer 96

W
Web links 21, 27
assigning to events 21
web servers 30, 87
WebFOCUS Maintain 15, 16, 17
benefits 17

WebFOCUS Maintain (continued)
n-tier applications 16
WebFOCUS Maintain advanced tutorial 99
WebFOCUS Maintain basic tutorial 33
WebFOCUS procedures 16, 27, 28, 30, 87, 135
deploying 30, 87
multiple copies of reports 135
WebFOCUS Servers 16, 30, 83, 87
deploying to 30, 87
WebFOCUS1.gif 99, 124
WinClose function 81
WinExit function 81
Winform command 66, 110
wizards 25, 26, 90, 114
Language Wizard 25, 26
Resource Wizard 90
Type Wizard 114
Customer Connections

Where users partner with Information Builders to provide feedback and create quality documentation.

Partner With Us

Information Builders Technical Content Management team is comprised of many talented individuals who work together to design and deliver quality technical documentation products. Your feedback supports our ongoing efforts!

You can also join a usability study to get an early look at new documentation products and services. Your participation helps us create documentation products that meet your needs.

To send us feedback or make a customer connection, contact Michele Brady, Service Director, Technical Content Management at Michele_Brady@ibi.com.

Request Permission to Use Copyright Material

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at Frances_Gambino@ibi.com.

Visit the Information Builders Technical Content Library

To visit the Technical Content Library, go to http://documentation.informationbuilders.com/. 
WebFOCUS 8
Maintain Getting Started
Release 8.1 Version 05