

Construct Spectrum SDK for Web Applications

Manual Order Number: SPV451-022IBW

This document applies to Construct Spectrum SDK Version 4.5 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Readers' comments are welcomed. Comments may be addressed to the Documentation Department at the address on the back cover or to the following e-mail address:

Documentation@softwareag.com

© Copyright Software AG 2003

All rights reserved

Printed in the Federal Republic of Germany

The name Software AG and/or all Software AG product names are either trademarks or registered trademarks of Software AG. Other company and product names mentioned herein may be trademarks of their respective owners.

TABLE OF CONTENTS

PREFACE

Prerequisite Knowledge.	10
Structure of this Documentation	11
How to Use this Documentation	13
Create a Complete Web Application.	13
Create the Web Components Only	14
Customize and Regenerate Application Components.	14
Document Conventions	15
Other Resources.	16
Related Documentation	16
Construct Spectrum SDK	16
Construct Spectrum	17
Natural Construct.	17
Other Documentation	18
Related Courses.	18

1. INTRODUCTION

What is Construct Spectrum?	20
Related Tools.	20
Add-Ins to Visual Basic.	20
HTML Editor.	22
Supported Web Browsers	22
Web Server Management	22
Architecture of a Web Application	23
Internet/Intranet	23
Internet Information Server (IIS).	24
Mainframe Server	26
Overview of the Development Procedure	28
Step 1: Plan and Design Your Web Application.	29
Plan the Deployment and Operation	29
Plan the Development	29
Design the Web Application	30
Step 2: Set Up Your Application Environment.	30
Step 3: Generate the Natural Components	30
Step 4: Create an ABO Project	31
Step 5: Generate the ABOs	31

Step 6: Create a Spectrum Web Project	31
Step 7: Generate the Web Components.....	32
Step 8: Customize Your Application.....	32
Step 9: Test and Debug Your Application	32
Step 10: Deploy Your Application	33

2. FEATURES OF THE DEMO APPLICATION

Opening the Demo Web Applications.....	36
Features of the Home Page and Navigation Bar	37
Home Page.....	38
Navigation Bar.....	39
Login Page.....	40
Change Password Page	41
Administration Page	42
Debug Page	44
Frames Mode.....	46
Features of a Maintenance Page	47
Header	48
Sections	48
Collapsible Sections	48
View Options.....	49
Single Edit View (Default)	49
Single Edit with Report View	49
Multiple Edit View	49
Multiple Edit Text Area View.....	49
Find Buttons	50
Calendar Pop-Up.....	50
Action Bar	51
Message Area	52
Links to Other Pages	52
Features of a Browse Page	53
Sort Key Selection.....	54
Multiple Sort Keys.....	54
Range Options	54
Go and More Buttons	54
Links to Other Pages	55

3. CREATING A WEB PROJECT	
Using the Spectrum Web Project Wizard	58
4. SUPPLIED FRAMEWORK COMPONENTS	
Introduction	64
Active Server Page (ASP) Script and Files	65
BAS (.bas) Files.	66
Cascading Style Sheet (.css) File.	67
HTML Templates and Page Handlers.	68
Examples of HTML Templates.	71
JavaScript (.js) Files	73
5. CREATING AND CUSTOMIZING A PAGE HANDLER	
Generating a Page Handler	76
Step 1: Invoke the Wizard	77
Step 2: Select an ABO.	78
Step 3: Confirm Details about the ABO	80
Step 4: Configure the Page Handler	81
Step 5: Generate the Page Handler	83
Customizing a Page Handler	84
Implied User Exits.	84
Supplied User Exits.	84
Content.CustomContentIDs.	84
BDT.Overrides	85
ParseTemplate.CustomTags	85
PerformAction.OtherResets.	86
PerformAction.CustomUpdateActions	86
PerformAction.UpdateForeignKeys	86
PerformAction.ClientValidations	87
PerformAction.CustomActions	87
RetrieveFromSession.CustomState and StoreToSession.CustomState.	88
UpdateData.CustomUpdates	88
Process.CustomActions	89
Modifying and Protecting Generated Code.	89

6. CREATING AND CUSTOMIZING AN HTML TEMPLATE

Introduction	92
Supplied HTML Templates	92
HTML Replacement Tags	92
Customizing HTML Templates	93
Before Generating the Template	93
After Generating the Template	93
Creating Custom Replacement Tags	93
Using Supplied Framework Components	93
Using the HTML Template Wizard	94
Step 1: Invoke the Wizard	95
Step 2: Select an ABO.	96
Step 3: Confirm ABO Details	97
Step 4: Configure the HTML Template	98
Step 5: Customize the HTML Template (Optional)	100
Step 6: Generate the HTML Template	100
After Generation is Complete	102
Customizing HTML Before Generation	103
Customizing a Maintenance Page	105
Deselect Fields for Generation	105
Change the Type of Control for a Field.	106
Specify or Modify Options on a Selection List.	107
Specify or Modify Options in a Radio Button Group	109
Change View Options for Sections	110
Single Edit View	110
Single Edit View with Report View Option	110
Multiple Edit View	111
Multiple Edit Text Area View	111
Collapsible Sections	111
Change View Options	112
Change the Width of a Text Box or Text Area	113
Change the Control Caption	113
Add a Link to a Browse Page	114
Add a Link to a Maintenance Page	117
Customizing a Browse Page	117
Deselect Fields for Generation	117
Change the Alignment of a Column in a Browse Table	118
Add a Link to a Maintenance Page	118
Add a Link to a Browse Page	120
Change Header Text	120

7. USING HTML REPLACEMENT TAGS

How Page Handlers Process Replacement Tags 122

Syntax for Replacement Tags 123

 Simple 123

 Conditional 123

 Repeating 124

 Complex 124

Supplied Replacement Tags 125

Defining Custom Replacement Tags. 135

 Add Code to the TagProcessing.bas File. 135

 Modify the ParseTemplate Function for the Page Handler 135

8. UPDATING AND CUSTOMIZING THE OBJECT FACTORY

Introduction 138

Using the Object Factory Wizard 139

User Exits in the Object Factory 142

 DefaultPage.SetDefault 142

 Security User Exits 142

9. VALIDATING YOUR DATA

Types of Validations Used in Web Applications 144

 Business Data Types (BDTs) 144

 Validation Routines in the ABO 144

 Validation Routines in the Page Handler 144

 Predict Verification Rules 145

 Cached Errors on the Debug Page. 146

10. SECURING YOUR APPLICATION

Using Construct Spectrum Security 150

 Spectrum Security 150

 Secure Sockets Layer (SSL) 150

 Secure Login Functionality 150

 Customizing Security 151

 Coding User Exits in the Object Factory. 151

 IsPermitted.Override 151

 ValidateUser.Override 151

 IsPermittedCustomTags. 151

 Customizing the Globals.bas File 152

Using Microsoft Internet Information Server Security 153

 Key Manager 153

 Server Certificate. 153

 SSL Key Pair. 153

11. DEPLOYING YOUR APPLICATION

Before Deploying the Application. 156
 Step 1: Create ActiveX/COM Dynamic Link Libraries (DLLs). 156
 Step 2: Register the Web DLL 157
 Step 3: Create a Virtual Directory (Optional) 157
 Step 4: Create a Starting Point for the Application. 157
 Step 5: Modify Application Settings. 158
Using the Package and Deployment Wizard. 159
 Before You Begin 159
 Create the Distributable Package. 159
 Deploy the Package. 160

12. TIPS WHEN USING OTHER PROGRAMMING LANGUAGES

Using Javascript. 162
 Using the Equal Sign to Test for Equality. 162
 Indicating Comments in Code. 162
 Substituting Variables 162
Using XSL. 163
 Evaluating the Contents of a Variable. 163
 Incrementing a Variable in an XSL Loop 163
 Updating a Web Page Automatically 163
Using HTML. 164
 Substituting Variables 164

APPENDIX A: GLOSSARY165

INDEX181

PREFACE

Welcome to *Construct Spectrum SDK for Web Applications*, designed to help developers create and customize the web components of applications using the Construct Spectrum software development kit (SDK) and Visual Basic.

This preface will help you get the most benefit from the documentation, as well as find other sources of information about creating Construct Spectrum web applications.

The following topics are covered:

- **Prerequisite Knowledge**, page 10
- **Structure of this Documentation**, page 11
- **How to Use this Documentation**, page 13
- **Document Conventions**, page 15
- **Other Resources**, page 16

Prerequisite Knowledge

This documentation does not provide information about the following topics. We assume that you are either familiar with the topics or have access to other sources of information about them.

- Natural Construct
- Microsoft Visual Basic
- Predict
- Natural programming language and environment
- Entire Broker
- Entire Net-Work

See **Other Resources**, page 16, for sources of information about Natural Construct and Construct Spectrum.

Structure of this Documentation

This section describes the contents of each chapter in this documentation. For information about how to use the documentation, see **How to Use this Documentation**, page 13.

The chapters in *Construct Spectrum SDK for Web Applications* are:

Chapter	Title	Topics
1	Introduction , page 19	Describes the components of Construct Spectrum and the architecture of the web applications you can create using the software development kit (SDK). It includes an overview of the procedure to develop a web application.
2	Features of the Demo Application , page 35	Describes the sample web application supplied with Construct Spectrum. This application demonstrates the features of a simple web application generated using Construct Spectrum.
3	Creating a Web Project , page 57	Describes how to use the Spectrum Web Project wizard to create a web project.
4	Supplied Framework Components , page 63	Describes the framework components supplied with Construct Spectrum. This set of generic, customizable components provide the basic structure and appearance of the application.
5	Creating and Customizing a Page Handler , page 75	Describes how to use the Page Handler wizard to generate page handlers for your web application. It includes information about using the user exits supplied in page handlers.
6	Creating and Customizing an HTML Template , page 91	Explains how to use the HTML Template wizard to create maintenance and browse pages.
7	Using HTML Replacement Tags , page 121	Describes the syntax of the HTML replacement tags, special replacement indicator tags embedded in the HTML templates that allow you to have “live” content presented programmatically on your web pages. It also describes the tags supplied with Construct Spectrum and how to create your own tags.

Chapter	Title	Topics (continued)
8	Updating and Customizing the Object Factory , page 137	Explains when, why, and how to update your web application's object factory using the Object Factory wizard. It also describes how you can use the user exits in the object factory to customize your web application.
9	Validating Your Data , page 143	Describes the data validation functionality provided with Construct Spectrum. It also explains how errors are displayed and handled in your web application.
10	Securing Your Application , page 149	Describes how to use the security facilities supported by Construct Spectrum and Microsoft Internet Information Server to secure your application. It also explains how to use the object factory's user exits and the Globals.bas module to modify security logic and functionality in Visual Basic.
11	Deploying Your Application , page 155	Describes how to deploy your application manually or using the Package and Deployment Wizard in Visual Basic. There are three methods you can use to deploy your web application: manually, using the Package and Deployment Wizard, or using third party software.
12	Tips When Using Other Programming Languages , page 161	Contains several tips and techniques you can consult when using other programming languages with the Construct Spectrum SDK for web applications.
Appendix A	Appendix A: Glossary , page 165	Contains definitions of terms used throughout the Construct Spectrum documentation set.

How to Use this Documentation

Construct Spectrum SDK for Web Applications describes how to create and customize the web components of an application, such as HTML templates, page handlers, and object factory entries. It also describes how to debug, secure, and deploy your web application. Use this documentation in combination with *Construct Spectrum SDK Reference* to:

- Create a complete web application
- Create the web components only
- Customize and regenerate application components

The following sections describe each of these options.

Note: Before you can generate and customize the web components, the Natural modules (subprograms, parameter data areas, and subprogram proxies) must exist on the mainframe and an ABO project containing ActiveX business objects must be available. The generated web components use the information in these objects to display and modify business data.

Create a Complete Web Application

To create a complete web application, including the Natural modules and ActiveX business objects, use this documentation and refer to the following chapters in *Construct Spectrum SDK Reference*:

- **Introduction**, page 21, for overviews of the product, the application architecture, and the development process.
- **Setting up the Mainframe Environment**, page 37, for information on defining domains and security options to control what data users can access on the mainframe.
- **Features of the Wizards**, page 49, for information about:
 - setting configuration options for the wizards
 - using Spectrum's client-side cache
 - modifying code frames
- **Using the Business-Object-Super-Model**, page 71, for information on using the super model wizard to generate the Natural components of your application.
- **Using ActiveX Business Objects**, page 85, for information about using the Construct Spectrum wizards to create ABOs and an ABO project.

Create the Web Components Only

If the Natural and ActiveX components of your application are currently available, *Construct Spectrum SDK for Web Applications* provides most of the information you need to create and customize web components and to deploy your web application.

For additional information, see the following chapters in *Construct Spectrum SDK Reference*:

- **Features of the Wizards**, page 49, for information about:
 - setting configuration options for the wizards
 - using Spectrum’s client-side cache
 - modifying code frames
- **Using Business Data Types (BDTs)**, page 121, for information about using BDTs in web applications.

Customize and Regenerate Application Components

As you customize and regenerate application components, use this documentation and refer to the following chapters in *Construct Spectrum SDK Reference* for useful information:

- **Using the Subprogram-Proxy Model**, page 103
- **Using Business Data Types (BDTs)**, page 121

Document Conventions

This documentation uses the following typographical conventions:

Example	Description
Introduction	Bolded text in cross references indicates chapter and section titles.
“A”	Items within quotation marks indicate values you must enter.
Browse model, GotFocus, Enter	Mixed case text indicates names of: <ul style="list-style-type: none"> • Natural Construct and Construct Spectrum editors, fields, files, functions, models, panels, parameters, subsystems, variables, and dialogs • Visual Basic classes, constants, controls, dialogs, events, files, menus, methods, properties, and variables • Keys
Alt+F1	A plus sign (+) between two key names indicates that you must press the keys together to invoke a function. For example, Alt+F1 means hold down the Alt key while pressing the F1 key.
CHANGE-HISTORY	Uppercase text indicates the names of Natural command keywords, command operands, data areas, help routines, libraries, members, parameters, programs, statements, subprograms, subroutines, user exits, and utilities.
<i>Construct Spectrum SDK for Web Applications, variable name</i>	Italicized text indicates: <ul style="list-style-type: none"> • Book titles • Placeholders for information you must supply
[<i>variable</i>]	In syntax and code examples, values within square brackets indicate optional items.
{WHILE UNTIL}	In syntax examples, values within brace brackets indicate a choice between two or more items; each item is separated by a vertical bar ().

Other Resources

This section provides information about other resources you can use to learn more about Construct Spectrum and Natural Construct. For more information about these documents and courses, contact the nearest Software AG office or visit the website at www.softwareag.com to order documents or view course schedules and locations. You can also use the website to email questions to Customer Support.

Related Documentation

This section lists other documentation in the Construct Spectrum and Natural Construct documentation set.

Construct Spectrum SDK

- *Construct Spectrum SDK for Microsoft .NET Framework*
This documentation is for developers creating Web services to invoke Natural subprograms (business objects) over the Inter/Intranet via the W3C SOAP standard.
- *Construct Spectrum SDK for Client/Server Applications*
This documentation is for developers creating client components for applications that will run in the Natural mainframe environment (server) and a Windows environment (client).
- *Construct Spectrum Messages*
This documentation is for application developers, application administrators, and system administrators who want to investigate messages returned by Construct Spectrum runtime and SDK components.
- *Construct Spectrum SDK Reference*
This documentation is for developers creating Natural modules and ActiveX Business Objects to support applications that will run in the Natural mainframe environment and a Windows environment and/or an internet server.

Construct Spectrum

- *Construct Spectrum Administration*
This documentation is for administrators who want to use the Construct Spectrum Administration subsystem to set up and manage Construct Spectrum applications.
- *Construct Spectrum Reference*
This documentation is for application developers and administrators who need quick access to information about Construct Spectrum application programming interfaces (APIs) and utilities.
- *Construct Spectrum and SDK Vn Release Notes*
These notes contain information on new features, enhancements, and other changes in this version of Construct Spectrum.
- *Construct Spectrum and SDK Installation Guide for Windows*
This guide describes how to install and set up the Construct Spectrum runtime and SDK components on the client.
- *Construct Spectrum and SDK Installation Guide for Mainframes*
This guide describes how to install and set up the Construct Spectrum runtime and SDK components on the mainframe.

Natural Construct

- *Natural Construct Installation Guide for Mainframes*
This guide provides essential information for setting up the latest version of Natural Construct, which is needed to operate the Construct Spectrum programming environment.
- *Natural Construct Generation*
This documentation describes how to use the Natural Construct models to generate applications that will run in a mainframe environment.
- *Natural Construct Administration and Modeling*
This documentation describes how to use the Administration subsystem of Natural Construct and how to create new models.
- *Natural Construct Help Text*
This documentation describes how to create online help for applications that run on server platforms.
- *Natural Construct Getting Started Guide*
This guide introduces new users to Natural Construct and provides step-by-step instructions to create several common processes.

Other Documentation

This section lists documents published by WH&O International:

- *Natural Construct Tips & Techniques*
This book provides a reference of tips and techniques for developing and supporting Natural Construct applications.
- *Natural Construct Application Development User's Guide*
This guide describes the basics of generating Natural Construct modules using the supplied models.
- *Natural Construct Study Guide*
This guide is intended for programmers who have never used Natural Construct.

Related Courses

In addition to the documentation, the following courses are available from Software AG:

- A self-study course on Natural Construct fundamentals
- An instructor-led course on building applications with Natural Construct
- An instructor-led course on modifying the existing Natural Construct models or creating your own models

INTRODUCTION

This chapter describes the components of Construct Spectrum and the software development kit (SDK). It describes the architecture of web applications you can create using the SDK and includes an overview of the development procedure.

Note: Detailed information about creating the web components in this procedure are described in later chapters; information about other steps is provided in *Construct Spectrum SDK Reference*.

The following topics are covered:

- **What is Construct Spectrum?**, page 20
- **Architecture of a Web Application**, page 23
- **Overview of the Development Procedure**, page 28

What is Construct Spectrum?

Construct Spectrum and the SDK comprise a set of middleware and framework components, as well as integrated tools, that use the specifications you supply to generate the components of a web application. These components include:

- Natural modules that allow you to call subprograms remotely
- ActiveX business objects that facilitate communication between the client and the mainframe server
- HTML pages that run in a browser to present business data to users

Note: This documentation provides information about generating and customizing the web components of a web application: page handlers, HTML templates, and object factory entries. For information about creating the Natural modules or ActiveX business objects, see *Construct Spectrum SDK Reference*.

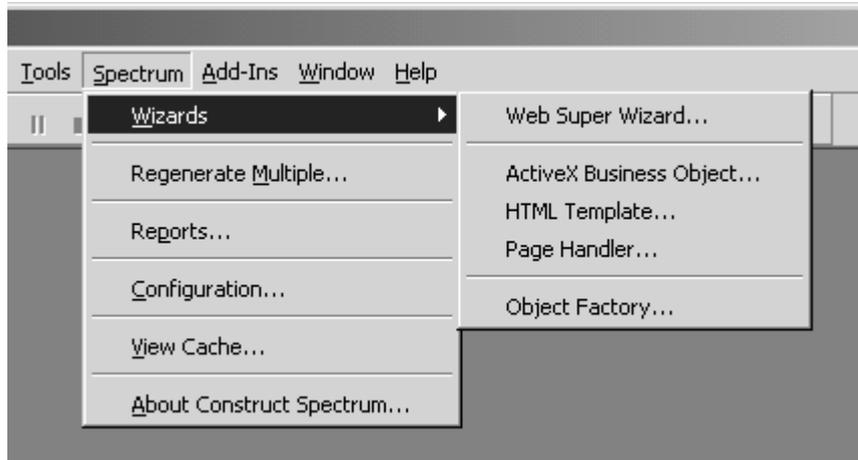
Related Tools

To create the web components of an application, you will work with several development and design tools. The following sections describe these tools.

Add-Ins to Visual Basic

Construct Spectrum supplies several wizards and utilities to help you develop the web components of your application. These tools are integrated into the Visual Basic development environment as Add-Ins and are available from the menu bar, as well as when you add a new web or ABO project.

After installing Construct Spectrum and opening Visual Basic, select the Spectrum Add-In. The Spectrum menu is displayed:



Spectrum Menu in Visual Basic

Use this menu to:

- Access wizards to generate the web components, such as ABOs, HTML templates, page handlers, and object factory entries; you can generate the components individually or use the Web Super wizard to generate all components
- Regenerate individual or multiple components
- View generation reports and compare code
- View and modify environmental options in the Configuration editor
- View and set options in the Spectrum cache

For information about using the Regenerate Multiple, Reports, Configuration, and View Cache options, see **Features of the Wizards**, page 49, *Construct Spectrum SDK Reference*.

HTML Editor

In addition to using Visual Basic's development environment to customize your HTML templates, you can use any HTML editor. For example:

- Microsoft Visual InterDev
- Microsoft FrontPage
- Homesite

If you are designing web pages for use with different web browsers, use an editor that creates generic HTML, rather than code that is suited to a particular browser.

Tip: Edit pages by changing the HTML directly, rather than dragging and dropping elements in WYSIWYG view. Construct Spectrum applications use HTML replacement tags at runtime to replace content dynamically and lay out page elements. As these tags are not visible in WYSIWYG, you may inadvertently change their order.

Supported Web Browsers

You can run and test your Construct Spectrum web application using the following versions of Microsoft Internet Explorer or Netscape Navigator:

- If you are not using fly-out menus in your web application, use Internet Explorer V5.0 (or higher) or Navigator V4.7 (or higher).
- If you are using fly-out menus, use Internet Explorer V5.5 (or higher) or Navigator V6.0 (or higher).

Construct Spectrum-generated web applications contain alternate functionality to accommodate different browsers. For example, field errors are highlighted when viewed in Internet Explorer. In Navigator, errors are indicated by a graphic.

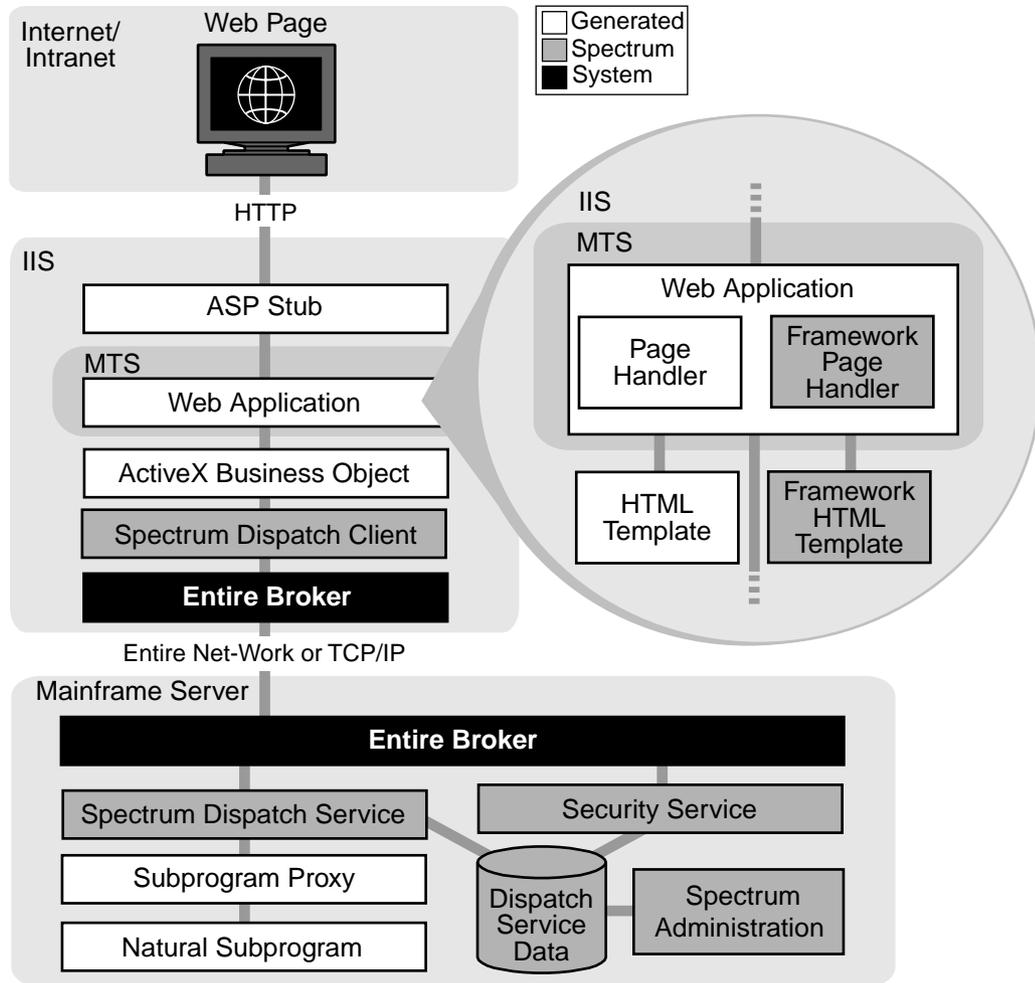
Web Server Management

You can use one of the following tools to manage the web server:

- If you are developing web applications on Windows NT, use Microsoft Management Console (MMC) to work with the Microsoft Internet Information Server (IIS) and administer the Microsoft Transaction Server (MTS), which runs the deployed Spectrum web application.
- If you are developing web applications on Windows 2000 or XP, use the Component Services manager and the Internet Services Manager under Administrative tools.

For information about MMC, IIS, MTS, and Personal Web Server, refer to the documentation supplied by Microsoft for these products.

Architecture of a Web Application



Architecture of a Construct Spectrum Web Application

The following sections describe the supported browsers, as well as the web components running on the internet information server (IIS) and mainframe server.

Internet/Intranet

Construct Spectrum web applications support both Microsoft Internet Explorer and Netscape Navigator browsers. For information about which versions are supported, see **Supported Web Browsers**, page 22.

Internet Information Server (IIS)

Construct Spectrum web applications work with the internet information server (IIS). The following components run on IIS:

Component	Description
Entire Broker	Transfers messages between the web server and the Natural environment. Entire Broker can be configured to use either native TCP/IP or Entire Net-Work as the transport layer.
Spectrum Dispatch Client (SDC)	<p>Component object model (COM) middleware component that enables the web applications to read and write variables in a Natural parameter data area (PDA) and to issue CALLNAT statements to Natural subprograms. Its main functions are:</p> <ul style="list-style-type: none"> • simulating PDAs and CALLNATs • encapsulating Entire Broker calls • controlling database transactions <p>As the client counterpart of Spectrum dispatch services, the SDC is also responsible for such things as data marshaling, encryption, compression, error-handling, and all Entire Broker communication.</p> <p>For information, see Using the Spectrum Dispatch Client, page 193, <i>Construct Spectrum SDK Reference</i>.</p>
ActiveX Business Object	<p>Object that encapsulates all communication with the Spectrum Dispatch Client and makes it efficient to invoke Natural services from the client. Each back-end business object is represented on the web server as an ActiveX business object.</p> <p>For information, see Using ActiveX Business Objects, page 85, <i>Construct Spectrum SDK Reference</i>.</p>
MTS	<p>Microsoft transaction server (MTS) that provides a controlled environment from which to run COM web components. MTS improves the scalability of applications, allowing them to expand to meet future needs. It also supplies a centralized facility for shutting down processes and updating the ActiveX dynamic link libraries (DLLs) that contain web applications. MTS also enables active server page (ASP) functionality in your web applications.</p>
Web application	<p>Combination of framework components supplied with all Construct Spectrum web projects and components that you create using Construct Spectrum wizards: HTML templates, page handlers, and object factory entries. The framework components include standard HTML templates, help routines, and the ASP files required to run your web application.</p>

Component	Description (continued)
Page handler	<p>Visual Basic class that manages web requests and responses associated with programmatically supplied HTML content.</p> <p>For information, see Creating and Customizing a Page Handler, page 75.</p>
HTML template	<p>Combination of HTML and JavaScript that presents a web page or a component of a web page in the browser. To build a web page, a number of Construct Spectrum components assemble and process the HTML templates and send the resulting HTML to the browser.</p> <p>Templates can contain placeholders for information that is retrieved at runtime, such as database information. One of the supplied framework components is a template parser that parses the starting template and resolves references to all sub-templates. Templates are processed until all placeholder information has been replaced. The resulting HTML is sent to the browser as an HTTP response.</p> <p>For information, see Creating and Customizing an HTML Template, page 91.</p>
Framework page handler and HTML template	<p>Page handlers and HTML templates supplied for the navigation bar, header, footer, login page, etc. You can customize these components for your application. The framework also supplies utilities and help routines.</p> <p>For information, see Supplied Framework Components, page 63.</p>
ASP Stub	<p>Active server page (ASP) script used to activate a specific web page. This script passes information obtained from the web server, such as session and application objects, to a web application associated with the ASP script.</p> <p>For information, see Active Server Page (ASP) Script and Files, page 65.</p>

Mainframe Server

The following components run on the mainframe server:

Component	Description
Natural subprograms	Subprograms that perform maintenance and browse functions on the mainframe. Construct Spectrum web applications communicate with subprograms created for the application, as well as existing Natural subprograms. For information about creating subprograms for a web application, see Using the Business-Object-Super-Model , page 71, <i>Construct Spectrum SDK Reference</i> .
Subprogram proxy	Bridge between a subprogram and the Spectrum dispatch service. The subprogram proxy performs a number of functions, including: <ul style="list-style-type: none"> translate parameter data into a format that can be transmitted between a web page and mainframe server issue CALLNATs to subprograms validate the format and length of data received from the client For information, see Using the Subprogram-Proxy Model , page 103, <i>Construct Spectrum SDK Reference</i> .
Spectrum dispatch service	Ensures that the current user is allowed to perform the requested function. Once the service authenticates the user ID, it activates the Natural subprogram that handles the requested function. After the subprogram completes its processing, the results are transferred back to the client. Depending on user options, the service can also compress and decompress and/or encrypt and decrypt messages. For information, see Defining and Managing Construct Spectrum Services , page 39, <i>Construct Spectrum Administration</i> .
Dispatch service data	Information defined and maintained in the Construct Spectrum Administration subsystem. Spectrum dispatch services use Entire Broker to access this information anywhere on the network.
Spectrum administration	Mainframe subsystem that allows system administrators, application administrators, and application developers to set up and manage system and application environments. For information, see <i>Construct Spectrum Administration</i> .

Component	Description (continued)
Security service	Spectrum security service that checks client requests against the security settings defined in the Construct Spectrum Administration subsystem. This stand-alone service operates independently of any one Spectrum dispatch service. This independence allows the security service to process, in one central location, the requests of several Spectrum dispatch services located on nodes throughout the network. For information about security services and security settings, see <i>Construct Spectrum Administration</i> .
Entire Broker	Transfers messages between the web server and the Natural environment. Entire Broker can be configured to use either native TCP/IP or Entire Net-Work as the transport layer.

Overview of the Development Procedure

This section contains an overview of the steps involved in developing a Construct Spectrum web application.

- To develop a Construct Spectrum web application:
 - ❑ **Step 1: Plan and Design Your Web Application**, page 29
 - ❑ **Step 2: Set Up Your Application Environment**, page 30
 - ❑ **Step 3: Generate the Natural Components**, page 30
 - ❑ **Step 4: Create an ABO Project**, page 31
 - ❑ **Step 5: Generate the ABOs**, page 31
 - ❑ **Step 6: Create a Spectrum Web Project**, page 31
 - ❑ **Step 7: Generate the Web Components**, page 32
 - ❑ **Step 8: Customize Your Application**, page 32
 - ❑ **Step 9: Test and Debug Your Application**, page 32
 - ❑ **Step 10: Deploy Your Application**, page 33

These steps are presented sequentially, but the process is an iterative one, especially when customizing, testing, and debugging your application. The following sections describe these steps and include cross-references to more information (when available).

Note: This documentation provides detailed information on steps 1 and 6–10. Steps 2–5 are described in *Construct Spectrum SDK Reference*.

Step 1: Plan and Design Your Web Application

There are many factors to consider when planning and designing your web application. Spending time on this phase of development will help you implement your web project more efficiently and avoid major revisions to accommodate future changes. The following sections discuss some of the planning and designing issues.

Plan the Deployment and Operation

The scope of the application is perhaps the most important consideration:

- Will the application be accessible to users on an intranet or the internet?
 - If your application will be used in a corporate setting via an intranet, issues of scalability and security may not be as important as they are when you are designing a web application for the internet.
 - If your application will be used on the internet, what security model is appropriate? For information, see **Securing Your Application**, page 149.
- How many users do you expect for this application? Can your web server handle that number? How will mainframe and web servers perform as the number grows?
- How long will it take to download graphics and other objects? Users will appreciate smaller downloadable files, especially if the application is accessed on the internet.
- What web browsers are users likely to employ? If you are creating an application for an intranet, the type and version of browser may be standardized for users.

Plan the Development

Will web applications be developed by a team of two or 20? If you are planning a large project that requires a multi-discipline team (perhaps comprising Natural and Visual Basic developers, web designers, web masters, system administrators, and technical communicators), consider the following points:

- Provide a separate internet server to help web designers test their pages.
- Choose a standard HTML editor for all members of the team to ensure that code is formatted consistently.

Design the Web Application

To help you design web sites and pages that are consistent, usable, and easy to navigate, Construct Spectrum provides the following:

- Framework HTML templates, cascading style sheets, and code frames you can customize. These framework components provide a starting point for building a unique and consistent corporate style for your applications. For example, you can modify the `Layout Template.htm` file so that your corporate logo appears in the same location on each web page that you generate.
- HTML replacement tags you can embed in HTML templates. These tags are replaced with actual values at runtime and provide flexibility in designing your web pages.

Step 2: Set Up Your Application Environment

Before creating a new web application, you must ensure that the application can access the correct Natural business objects — and that users can access the application.

➤ To set up your application environment:

- 1 Set up the Predict definitions.
Construct Spectrum works with Predict, a data dictionary and repository that manages metadata about information in the database your applications use. Predict “views” of data and the relationships between data help you define the business objects your applications access and maintain. Predict verification rules and keywords validate and format the data, and its field definitions automatically select controls for your applications. You can also use Predict to define the defaults used to generate your applications.
- 2 Define a steplib chain and domain in the Construct Spectrum Administration Subsystem and associate the steplib chain with the domain.
- 3 Set up security privileges for the domain by setting up users and groups and granting them access to the domain, its objects and methods.

For information, see **Setting up the Mainframe Environment**, page 37, *Construct Spectrum SDK Reference*.

Step 3: Generate the Natural Components

For each business object in your application (for example, Customer, Order, or Product), you must create subprograms, parameter data areas (PDAs), and subprogram proxies to implement maintenance and browse functions. Use the Business-Object-Super-Model to generate these components for up to 12 business object “packages” at a time. For information, see **Using the Business-Object-Super-Model**, page 71, *Construct Spectrum SDK Reference*.

Step 4: Create an ABO Project

Each Natural business object in your application is represented on the web server by an ActiveX business object (ABO). Many related ABOs are packaged into a single dynamic link library (DLL). This makes it easy to add a reference to several objects at one time during development. If the business objects are shared among web and client/server applications, combining multiple objects within a single DLL also simplifies the deployment of these objects.

Construct Spectrum provides the ABO project wizard to create an ABO project, which you can later compile into DLLs. For information, see **Features of the Wizards**, page 49, *Construct Spectrum SDK Reference*.

Step 5: Generate the ABOs

These COM objects represent Natural business objects. ABOs make it convenient for client components to interface with server objects using standard interfaces. They are a bridge between the client and server functions.

Construct Spectrum provides the ActiveX Business Object wizard to define and generate ABOs. For information, see **Using ActiveX Business Objects**, page 85, *Construct Spectrum SDK Reference*.

Step 6: Create a Spectrum Web Project

After creating the ABO project and populating it with the ABOs for your application, the next step is to create a Spectrum web project to contain the class modules, HTML templates, and support files for your application.

Construct Spectrum provides the Web Project wizard to generate the web project. This wizard is available through File > Add Project in the ABO project window. For information, see **Creating a Web Project**, page 57.

When generation is complete, several framework components are added to the web project.

Step 7: Generate the Web Components

Use the Web Super wizard to quickly generate the web components for each ABO. You can then customize or regenerate components using their respective wizards:

- **Page handlers**
To process a web page and replace tags, each HTML template requires a page handler. For more information, see **Creating and Customizing a Page Handler**, page 75.
- **HTML templates**
You must also create HTML files for each page in your application. For more information, see **Creating and Customizing an HTML Template**, page 91.
- **Object Factory entries**
Construct Spectrum web applications use a Visual Basic module, called an object factory, to encapsulate all the business objects used by an application. When you add or modify an ABO or page handler, use the Object Factory wizard to update the object factory. For information, see **Updating and Customizing the Object Factory**, page 137.

Note: To regenerate or customize Web Super wizard-generated web components, use the individual wizards (for example, the HTML Template or Page Handler wizard).

Step 8: Customize Your Application

After generating the basic modules for your web application, you can customize the HTML templates, ABOs, cascading style sheets, and framework components to present and process data to your specifications. You may have to use the regeneration facilities to regenerate modules. For information, see:

- **Creating and Customizing a Page Handler**, page 75.
- **Creating and Customizing an HTML Template**, page 91.
- **Supplied Framework Components**, page 63.

Step 9: Test and Debug Your Application

While developing an application, test the functionality by running it in a supported browser. For more information, see **Supported Web Browsers**, page 22.

Use the following tools to debug the application:

- To resolve errors, use the online Visual Basic debugging facilities. For information about these facilities, see the Microsoft Visual Basic documentation.
- To trouble-shoot communication problems between the client and server, use the Debug page supplied with Construct Spectrum. For information, see **Cached Errors on the Debug Page**, page 146.

Step 10: Deploy Your Application

- To deploy your application:
 - 1 Create dynamic link library (DLL) files for your ABO and Spectrum web projects.
 - 2 Package the application files.
 - 3 Install the application files on an internet server.
 - 4 Configure the application settings.

For information, see **Deploying Your Application**, page 155.

FEATURES OF THE DEMO APPLICATION

This chapter describes the sample web application supplied with Construct Spectrum. The application demonstrates the features of a simple web application generated using Construct Spectrum. Explore the demo application to become familiar with the components and functionality you can use in your own applications.

The demo is an Order Entry application that allows users to browse and maintain four business objects: Customer, Order, Product, and Warehouse. Using Predict files installed in the demo library, the application retrieves and updates data on the mainframe.

This chapter uses the Order Maint and Customer browse pages as seen in Microsoft Internet Explorer to illustrate the features and functions of the demo application. Some features of these pages are provided by default with every web application developed using Construct Spectrum; others are supplied based on Predict settings; others are the result of adding custom code.

The following topics are covered:

- **Opening the Demo Web Applications**, page 36
- **Features of the Home Page and Navigation Bar**, page 37
- **Features of a Maintenance Page**, page 47
- **Features of a Browse Page**, page 53

Opening the Demo Web Applications

The demo web applications are installed with the web components of Construct Spectrum SDK. By default, the installation process places them in the publishing path of your web server. (For example, the order entry demo web application is installed into `Inetpub\wwwroot\WebOrderEntry` if using Personal Web Server.) For information about installing and configuring the SDK, see *Construct Spectrum and SDK Installation Guide for Windows*.

As a result of the installation and configuration processes, a Spectrum dispatch service is now available on the web server.

Note: If you are using Entire Net-Work, ensure that an Entire Net-Work kernel is running on your PC before opening the demo application.

➤ To open a demo web application:

- 1 Open your browser: Internet Explorer or Navigator.
For more information, see **Supported Web Browsers**, page 22.
- 2 Provide one of the following URLs:

```
http://localhost/SpectrumWebDemos/weborderentry/cstorderentrydemo/  
webapp.asp
```

or

```
http://localhost/SpectrumWebDemos/NoahsPetSupplies/webapp.asp
```

where `localhost` is the name of your PC or “local host”.

The Home page of the specified demo application is displayed.

Note: To use the Noah’s Pet Supplies demo application, you must have MDAC (Microsoft Data Access Components) V2.5 or higher installed on your PC.

Note: For a link to all demo web applications, refer to:
<http://localhost/SpectrumWebDemos/Default.htm>

The following section describes features of the Home page and navigation bar, as well as how to login to the application and use the Administration page to view dispatch services. Illustrations are from a Microsoft Internet Explorer environment.

Features of the Home Page and Navigation Bar

The Home page is the starting point for the application — it is the first page you see when you open the Order Entry demo. The navigation bar, which is present on all web pages, gives you access to the:

- Business objects:
 - Customer Maint and Browse pages
 - Order Maint and Browse pages
 - Product Maint and Browse pages
 - Warehouse Maint and Browse pages
- Framework components:
 - Login page
 - Change Password page
 - Administration page
 - Debug page
 - Frames mode page

The following sections describe the Home page, the navigation bar, and the framework components listed above. For a description of the maintenance and browse pages for business objects, see **Features of a Maintenance Page**, page 47, and **Features of a Browse Page**, page 53.

Home Page

When you open the Order Entry demo application, the Home page is displayed in the browser:



Order Entry Demo — Home Page

The Home page provides introductory information and links to other pages in the web application, such as the Order Maint page or the Login page.

Construct Spectrum SDK supplies framework components to create a Home page for your application. To customize the Home page, modify the Home.htm HTML template and the Home.cls page handler in the web application directory. For information, see *Construct Spectrum SDK Reference*.

Navigation Bar

This bar, located along the left edge of the Home page shown on the previous page, provides links to other web pages and is available on every page in the demo application. To display a page, click the page name on the navigation bar.

When a user opens a web application, the navigation bar shows the business objects and standard options: Login, Change Password, Admin, Home, Debug, and Frames in the demo application. After the user opens the Login page and logs into the application, the navigation bar is populated with the maintenance and browse functions (methods) supported for that user. Security settings determine which objects and methods are available.

The navigation bar is another framework component supplied with Construct Spectrum SDK. To customize this component, modify the NavBar.htm HTML template and the NavBar.cls page handler in the web application directory. For information, see **Customizing HTML Templates**, page 93, and **Customizing a Page Handler**, page 84.

The following sections describe the standard pages accessed from the navigation bar in the demo application.

Login Page

- To login to the demo application and display the available business objects:
 - 1 Click Login on the navigation bar.
The Login page is displayed:



Order Entry Demo — Login Page

- 2 Enter your user ID.
- 3 Click Login.

The demo application accepts any user ID and a blank password.

If the call to the mainframe server is successful, the Home page is redisplayed and the navigation bar contains the business objects you can access and the methods available. In the demo application, for example, the Customer business object has links to Maint and Browse pages.

If the login was not successful, the error is described in the message area. If the problem was caused by an incorrect dispatch service, see **Administration Page**, page 42.

Once you have logged in, the Login option on the navigation bar changes to Logout. When you select Logout, the navigation bar options return to pre-authenticated status.

Note: You are logged out automatically if you do not use the application within the session timeout setting. To modify the timeout setting, use the Microsoft Management Console.

The Login page uses HTTP security to authenticate the user's ID and password. The first time a user logs in, the user ID is saved in a cookie. The ID is displayed on the Login page the next time the user logs in.

A Construct Spectrum-generated Login page can use a secure web protocol (HTTPS) while the rest of the application uses a normal unsecured protocol (HTTP). This ensures that a user's ID and password are encrypted when sent over the network during login. By default, web applications do not use this secure login functionality. To enable the option, modify the Globals.bas module. For information, see **Customizing the Globals.bas File**, page 152. For information about securing a web application, see **Securing Your Application**, page 149.

Construct Spectrum SDK supplies framework components to create a Login page for your application. To customize this page, modify the Login.htm HTML template and the Login.cls page handler in the web application directory. For information, see **Customizing HTML Templates**, page 93, and **Customizing a Page Handler**, page 84.

Note: When creating page handlers for your application, you can specify whether or not the user must login to your application. For information, see **Creating and Customizing a Page Handler**, page 75.

Change Password Page

The Change Password page allows users to change their password to a web application. (This option is not required for the demo application, as the application does not perform password checking.)

- To change your login password:
 - 1 Click Login on the navigation bar.
The Change Password page is displayed:



Order Entry Demo — Change Password Page

Construct Spectrum SDK supplies framework components to create a Change Password page for your application. To customize this page, modify the `ChangePassword.htm` HTML template and the `ChangePassword.cls` page handler in the web application directory. For information, see **Customizing HTML Templates**, page 93, and **Customizing a Page Handler**, page 84.

Administration Page

The Administration page allows users to view and change the Spectrum dispatch service used by the web server to access the mainframe server.

- To access the Administration page:
- 1 Click Admin on the navigation bar.
The Administration page is displayed:



Order Entry Demo — Administration Page

- 2 Select the dispatch service from the drop-down list box.
The Allow user to keep session data option is selected by default.
- 3 Click Save.
The current settings are saved.

The Allow user to keep session data option determines what happens to session data if you change the dispatch service during operation (for example, to access a different library or database). To allow users to keep working with current session data, this option must be marked. When a user makes a request to the server, the cached object is compared to the current dispatch service. If a difference is found, a message is displayed.

If the Allow user to keep session data option is not selected, the session data is discarded when the dispatch service changes, which may cause data to be lost.

When creating applications, you may want to limit access to the Admin page to users with Administration IDs, since the options on this page affect all users of the web application. For information about restricting the Admin link on the navigation bar, see **Securing Your Application**, page 149.

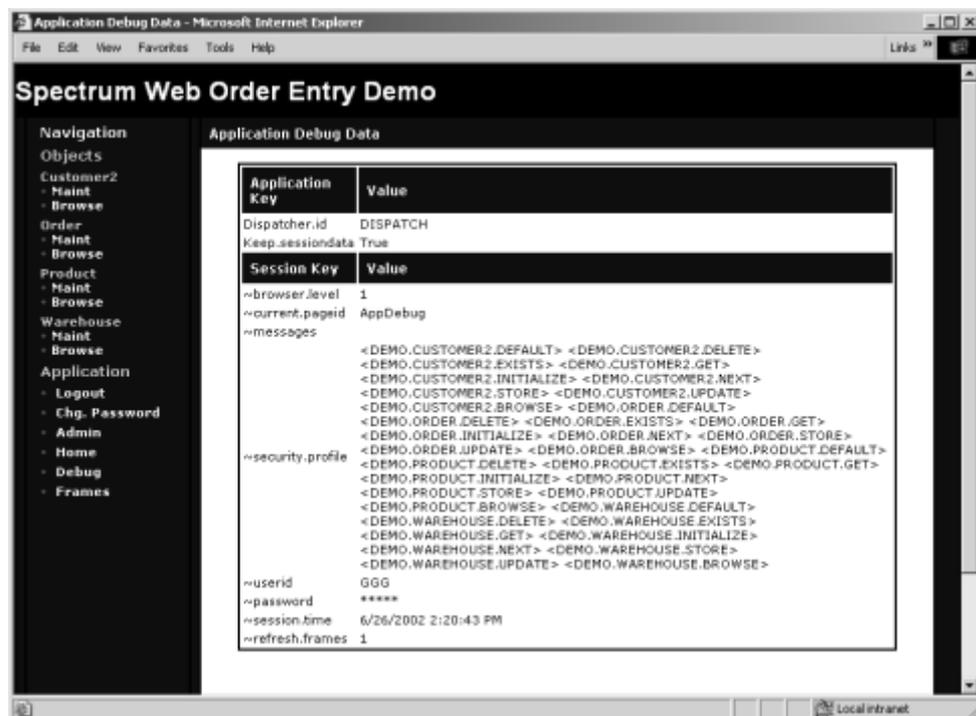
Construct Spectrum SDK supplies framework components to create an Admin page for your application. To customize this page, modify the Admin.htm HTML template and the Admin.cls page handler in the web application directory. For information, see **Customizing HTML Templates**, page 93, and **Customizing a Page Handler**, page 84.

Debug Page

The Debug page displays information about global performance and the current session, such as the objects accessed by the application and the cached errors.

Use this page when testing your web applications. Before distributing the application, either remove the page or retain it as a useful tool for support personnel in diagnosing problems.

- To access the Debug page:
 - 1 Click Debug on the navigation bar.
The Debug page is displayed:



Order Entry Demo — Debug Page

The Debug page displays the following information:

Key	Value
Application Key	Information that applies to the application, rather than an individual user's session. For example:
Dispatcher.id	Name of the dispatch service currently used to communicate with the mainframe.
Keep.sessiondata	Current value of the Allow users to keep session data option on the Administration page.
Session Key	Information specific to the user's current session, such as:
~browser.level	Web browser in use. 1 indicates Internet Explorer; 2 indicates Navigator.
~current.pageid	Name of the page handler for the Debug page.
~messages	Messages queued for display.
~security.profile	Access privileges for objects and methods in the current application. For example, <DEMO.CUSTOMER.GET> indicates the user is allowed to display a customer record. The syntax for these entries is domain.object.method. For information, see Setting up the Mainframe Environment , page 37, <i>Construct Spectrum SDK Reference</i> .
~userid	User ID used to logon to the application.
~password	User password used to logon. This text box is blank, since the demo application does not perform password checking.
	Note: If a password is used, it will not be displayed; it will be masked by asterisks (*).
~session.time	Date and time the user logged onto the application.
~refresh.frames	Frame refresh option (if Frames mode is on). 1 indicates frames are refreshed; 0 indicates frames are up to date.

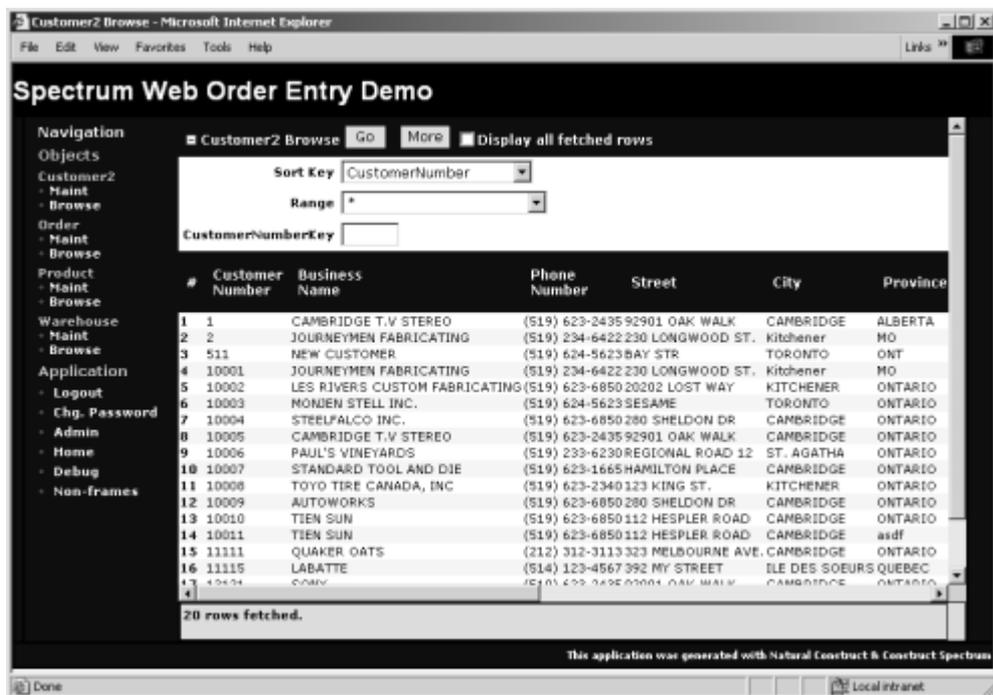
If a maintenance or browse page accesses mainframe data, the Debug page displays additional information (such as the cached errors). For information, see **Validating Your Data**, page 143.

Construct Spectrum SDK supplies framework components to create a Debug page for your application. To customize this page, modify the AppDebug.htm HTML template and the AppDebug.cls page handler in the web application directory. For information, see **Customizing HTML Templates**, page 93, and **Customizing a Page Handler**, page 84.

Frames Mode

The Frames mode option on the navigation bar allows users to toggle between Frames and Nonframes mode. If Frames mode is on, the page is divided into independent sections that scroll separately. For example, the heading and navigation bar remain stationary if the user scrolls down on the page.

The following example shows the Customer browse page in Frames mode:



Order Entry Demo — Frames Mode

Construct Spectrum supports this option in the supplied framework components for the Debug page. By default, NonFrames mode is on. To change the Frames option, modify the Frameset.htm HTML template and the Frameset.cls page handler in the web application directory. For information, see **Customizing HTML Templates**, page 93, and **Customizing a Page Handler**, page 84.

Features of a Maintenance Page

The Order Maint page in the demo application illustrates the functionality of a typical Construct Spectrum-generated maintenance page. It was created using the supplied framework components, as well as custom code. The supplied framework includes standard components used by all pages in the application, such as headers, as well as components that provide functionality specific to maintenance pages.

- To access the Order Maint page:
 - 1 Click the Maint link under Order on the navigation bar. The Order Maint page is displayed.
 - 2 Click Next to display data for the first order record in the file:



Order Maint Page

The following sections describe features of the Order Maint page that were created using the supplied framework components.

Note: For information on how the HTML templates and page handlers work together to present a page, see **HTML Templates and Page Handlers**, page 68.

Header

Spectrum Web Order Entry Demo

Header Framework Component

The header is another framework component supplied with Construct Spectrum. To customize the header, modify the Header.htm HTML template and the Header.cls page handler in the web application directory. For information, see *Construct Spectrum SDK Reference*.

Sections

Whenever a periodic group, multiple-valued field, or related file in a Predict view is specified, components of the group are generated into a table called a section. The name of the group becomes the section heading, as in Delivery Instructions and Order Lines on the Order Maint page. For example:



Index	Product ID	Quantity	Unit Cost	Total Cost
1	187261	10	\$150.00	\$1,500.00

Order Lines Section

When creating applications, you can modify the captions for sections and fields using the HTML Template wizard. For information, see **Customizing HTML Before Generation**, page 103.

Collapsible Sections

Web page sections are collapsible — users can toggle between seeing additional data and hiding the data by clicking the plus (+) and minus (-) icons. This option is useful to hide content the user does not currently need and it reduces the need for scrolling.

View Options

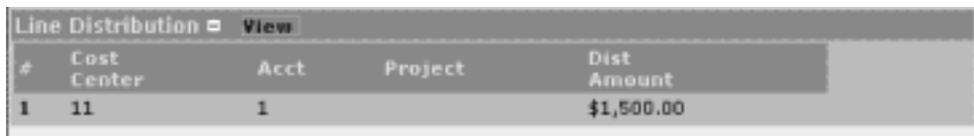
You can present section information in several views: single edit, single edit with report, multiple edit, or multiple edit text area. When creating applications, set the view options in the HTML Template wizard. For information, see **Customizing HTML Before Generation**, page 103.

Single Edit View (Default)

The Order Maint page in the demo application uses the default view, single edit. One record is displayed at a time and the user can modify all fields. For a related file, such as Order Lines, drop-down lists and action buttons are also available. The user can select and modify related records and then click Update.

Single Edit with Report View

When you define a section with single edit view, you can also allow the user to toggle between single edit view and report view by clicking View. For example:



Line Distribution	Cost Center	Acct	Project	Dist Amount
1	11	1		\$1,500.00

Report View

Multiple Edit View

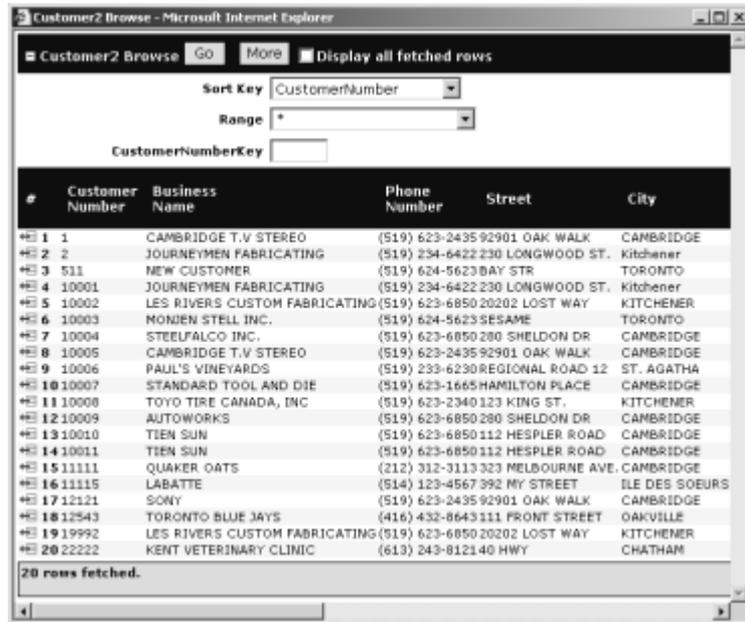
This view presents all records in a table that the user can edit.

Multiple Edit Text Area View

For alpha MU fields, you can present the section as a text area that users can edit.

Find Buttons

The Order Number, Customer Number, and Warehouse ID text boxes on the Order Maint page include a Find button. When you click Find, a browse page is displayed to select a row and return to the maintenance page. For example, the following page is displayed when you click Find for Customer Number:



Customer Browse Page

Click a red arrow to select the corresponding customer. The Order Maint page is redisplayed, showing the selected customer number in Customer Number.

The Find buttons in the demo application are the result of customizations before generating the HTML template. Clicking a Find button can display the information in the same window or in a pop-up window. For more information, see **Customizing HTML Before Generation**, page 103.

Calendar Pop-Up

On a maintenance page generated using the supplied framework components, Date information is automatically accompanied by a calendar pop-up. A user clicks the calendar icon to display the pop-up and select a date. To select the date, the user can use the pull-down list boxes to select a month and year and then click the day on the calendar.

Construct Spectrum SDK supplies a framework component to create a calendar pop-up for your application. To customize this page, modify the Calendar.htm HTML template in the web application directory. For information, see **Customizing HTML Templates**, page 93.

Action Bar

The methods available for a business object are displayed at the bottom of maintenance pages in the form of buttons on an action bar. For example:



Action Bar

The contents of the action bar vary, depending on user access privileges. For example, a user who can view orders but not update them will not see the Update, Add, or Delete buttons. For information, see **Securing Your Application**, page 149.

Construct Spectrum SDK supplies framework components to create an action bar for your application. To customize these components, modify the ActionBar.htm HTML template and the ActionBar.cls page handler in the web application directory. For information, see *Construct Spectrum SDK Reference*.

Message Area

Messages are displayed at the bottom of each page. For example, the following message indicates that 20 rows have been fetched from the database:

1	1	CAMBRIDGE T.V STEREO	(519) 623-2435 92901 OAK WALK	CAMBRIDGE
2	2	JOURNEYMEN FABRICATING	(519) 234-6422 230 LONGWOOD ST.	Kitchener
3	511	NEW CUSTOMER	(519) 624-5623 BAY STR	TORONTO
4	10001	JOURNEYMEN FABRICATING	(519) 234-6422 230 LONGWOOD ST.	Kitchener
5	10002	LES RIVERS CUSTOM FABRICATING	(519) 623-6850 20202 LOST WAY	KITCHENER
6	10003	MONDEN STELL INC.	(519) 624-5623 SESAME	TORONTO
7	10004	STEEFALCO INC.	(519) 623-6850 280 SHELDON DR	CAMBRIDGE
8	10005	CAMBRIDGE T.V STEREO	(519) 623-2435 92901 OAK WALK	CAMBRIDGE
9	10006	PAUL'S VINEYARDS	(519) 233-6230 REGIONAL ROAD 12	ST. AGATHA
10	10007	STANDARD TOOL AND DIE	(519) 623-1665 HAMILTON PLACE	CAMBRIDGE
11	10008	TOYO TIRE CANADA, INC	(519) 623-2340 123 KING ST.	KITCHENER
12	10009	AUTOWORKS	(519) 623-6850 280 SHELDON DR	CAMBRIDGE
13	10010	TIEN SUN	(519) 623-6850 112 HESPLER ROAD	CAMBRIDGE
14	10011	TIEN SUN	(519) 623-6850 112 HESPLER ROAD	CAMBRIDGE
15	11111	QUAKER OATS	(212) 312-3113 323 MELBOURNE AVE.	CAMBRIDGE
16	11115	LABATTE	(514) 123-4567 392 MY STREET	ILE DES SOEURS
17	12121	SONY	(519) 623-2435 92901 OAK WALK	CAMBRIDGE
18	12543	TORONTO BLUE JAYS	(416) 432-8643 111 FRONT STREET	OAKVILLE
19	19992	LES RIVERS CUSTOM FABRICATING	(519) 623-6850 20202 LOST WAY	KITCHENER
20	22222	KENT VETERINARY CLINIC	(613) 243-8123 40 HWY	CHATHAM

20 rows fetched.

Message Area

Construct Spectrum SDK supplies framework components to create a message area for your application. To customize this component, modify the Messages.htm HTML template and the Messages.cls page handler in the web application directory. For information, see *Construct Spectrum SDK Reference*.

Links to Other Pages

You can link a field on the maintenance page to another page in your web application. For example, you can link a field on a browse page to a related browse or maintenance page. If a field contains a link to another page, it is displayed in bold.

Use the HTML Template wizard to define links between web pages. For information, see **Customizing HTML Before Generation**, page 103.

Features of a Browse Page

The Customer Browse page in the demo application illustrates the functionality of a typical generated browse page. It was created using the supplied common framework components described in **Features of a Maintenance Page**, page 47, as well as other components that provide functionality specific to browse pages.

- To access the Customer Browse page:
 - 1 Click the Browse link under Customer on the navigation bar.
The Customer Browse page is displayed.
 - 2 Click Go.
The first 20 rows (records) are fetched:

The screenshot shows a web browser window titled "Customer2 Browse - Microsoft Internet Explorer". The page content includes a navigation menu on the left, a search form at the top, and a table of customer records. The search form has a "Sort Key" dropdown set to "CustomerNumber", a "Range" dropdown, and a "CustomerNumberKey" input field. The "Go" button is visible. Below the search form is a table with the following data:

#	Customer Number	Business Name	Phone Number	Street	City
1	1	CAMBRIDGE T.V STEREO	(519) 623-2435	92901 OAK WALK	CAMBRIDGE
2	2	JOURNEYMEN FABRICATING	(519) 234-6422	230 LONGWOOD ST.	KITCHENER
3	511	NEW CUSTOMER	(519) 624-5623	BAY STR	TORONTO
4	10001	JOURNEYMEN FABRICATING	(519) 234-6422	230 LONGWOOD ST.	KITCHENER
5	10002	LES RIVERS CUSTOM FABRICATING	(519) 623-6850	20202 LOST WAY	KITCHENER
6	10003	MONDEN STELL INC.	(519) 624-5623	SESAME	TORONTO
7	10004	STEEFALCO INC.	(519) 623-6850	280 SHELDON DR	CAMBRIDGE
8	10005	CAMBRIDGE T.V STEREO	(519) 623-2435	92901 OAK WALK	CAMBRIDGE
9	10006	PAUL'S VINEYARDS	(519) 233-6230	REGIONAL ROAD 12	ST. AGATHA
10	10007	STANDARD TOOL AND DIE	(519) 623-1665	HAMILTON PLACE	CAMBRIDGE
11	10008	TOYO TIRE CANADA, INC	(519) 623-2340	123 KING ST.	KITCHENER
12	10009	AUTOWORKS	(519) 623-6850	280 SHELDON DR	CAMBRIDGE
13	10010	TIEN SUN	(519) 623-6850	112 HESPLER ROAD	CAMBRIDGE
14	10011	TIEN SUN	(519) 623-6850	112 HESPLER ROAD	CAMBRIDGE
15	11111	QUAKER OATS	(212) 312-3113	323 MELBOURNE AVE.	CAMBRIDGE
16	11115	LABATTE	(514) 123-4567	392 NY STREET	ILE DES SOEURS
17	12121	SONY	(519) 623-2435	92901 OAK WALK	CAMBRIDGE
18	12543	TORONTO BLUE JAYS	(416) 432-8643	111 FRONT STREET	OAKVILLE
19	19992	LES RIVERS CUSTOM FABRICATING	(519) 623-6850	20202 LOST WAY	KITCHENER
20	22222	KENT VETERINARY CLINIC	(613) 243-8123	40 HWY	CHATHAM

At the bottom of the table, it says "20 rows fetched." The page also includes a "Display all fetched rows" checkbox and a "More" button.

Customer Browse Page

Because there are more records in the file, the More button is displayed, as well as the Display all fetched rows check box.

The following sections describe features of the Customer Browse page that were created using the supplied framework components.

Sort Key Selection

Construct Spectrum SDK supplies framework components to select the sort key for a browse page. To customize these components, modify the `KeySelector.htm` HTML template and the `KeySelector.cls` page handler in the web application directory. For information, see **Customizing HTML Templates**, page 93, and **Customizing a Page Handler**, page 84.

The `KeySelector` page handler queries the browse ABO to ascertain the browse key components. The `KeySelector` HTML template supports multiple sort keys and range options, presenting them in a drop-down list box for selection. The following sections describe these options.

Multiple Sort Keys

- To change the sort key:
 - 1 Select another sort key from the Sort Key drop-down list box.
 - 2 Click Go.
The first 20 rows are fetched and arranged according to the sort key selection.

Range Options

- To select a range option:
 - 1 Select a range indicator from the Range drop-down list box.
 - 2 Type the range criteria in the `CustomerNumberKey` field.
For example, you can select “>” from the Range list box and specify a customer number in this field. Only customer numbers greater than the specified number are displayed.
 - 3 Click Go.
The first 20 rows in the specified range are fetched.

Go and More Buttons

Construct Spectrum-generated browse pages include a Go button to retrieve up to 20 rows from the database for display on the browse page — one row for each record. If there are more than 20 records in the file, click the More button to retrieve the next 20 rows, etc. By default, only 20 records are displayed at one time. To display all fetched records, mark the Display all fetched rows check box.

Links to Other Pages

Construct Spectrum displays records in a table on the browse page and uses the field names as column headers. You can link a field on the browse page to another page in your web application. In the demo application, for example, you can click a Customer ID on the Customer Browse page to open the corresponding Customer Maint page. If a field contains a link to another page, it is displayed in bold.

Use the HTML Template wizard to define links between browse and maintenance pages. For information, see **Customizing HTML Before Generation**, page 103.

CREATING A WEB PROJECT

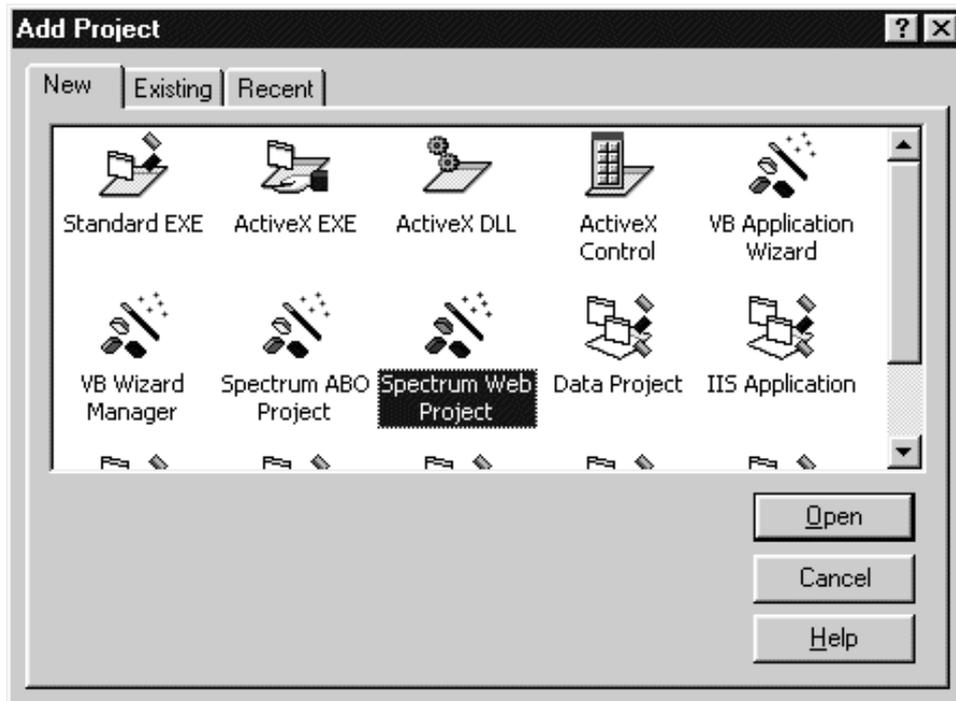
Before generating the web components for your application, you must have access to the ABO project and generated ABOs. In addition, you must create a Construct Spectrum web project to contain the generated components. This chapter describes how to use the Spectrum Web Project wizard to create your web project.

For information on creating an ABO project and generating the ABOs, see **Using ActiveX Business Objects**, page 85, *Construct Spectrum SDK Reference*.

Using the Spectrum Web Project Wizard

The web project contains the page handlers, HTML templates, and other framework components for your web application.

- To create a web project:
 - 1 Open the existing ABO project or the project group for the application.
 - 2 Select Add Project from the File menu.
The Add Project window is displayed:

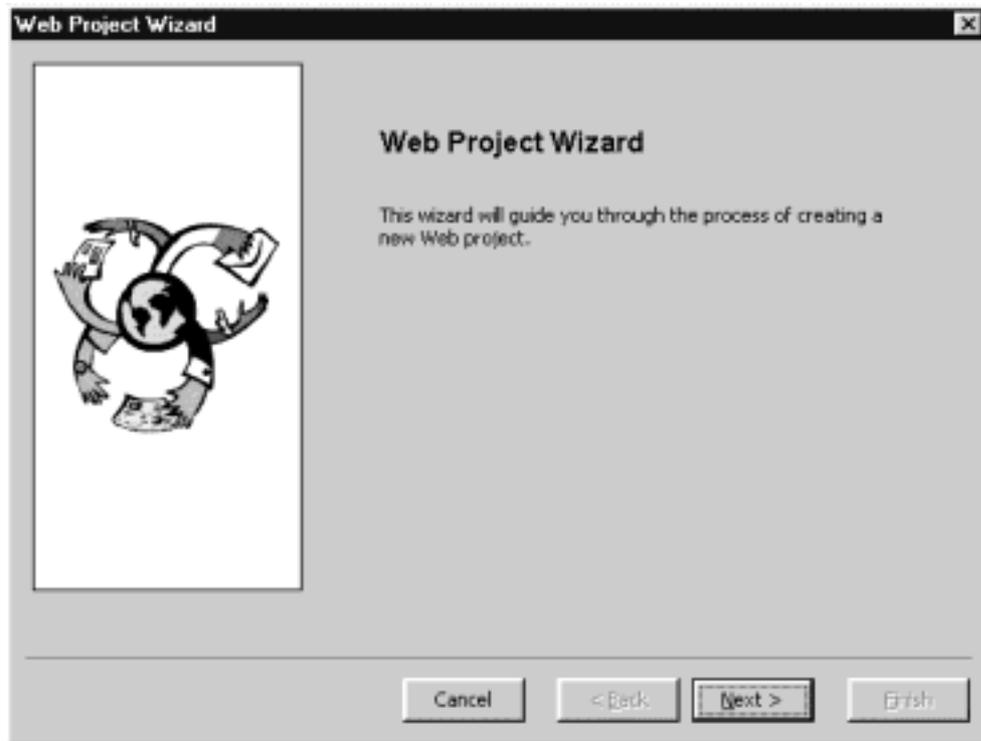


Add Project Window — New Tab

The New tab is displayed by default.

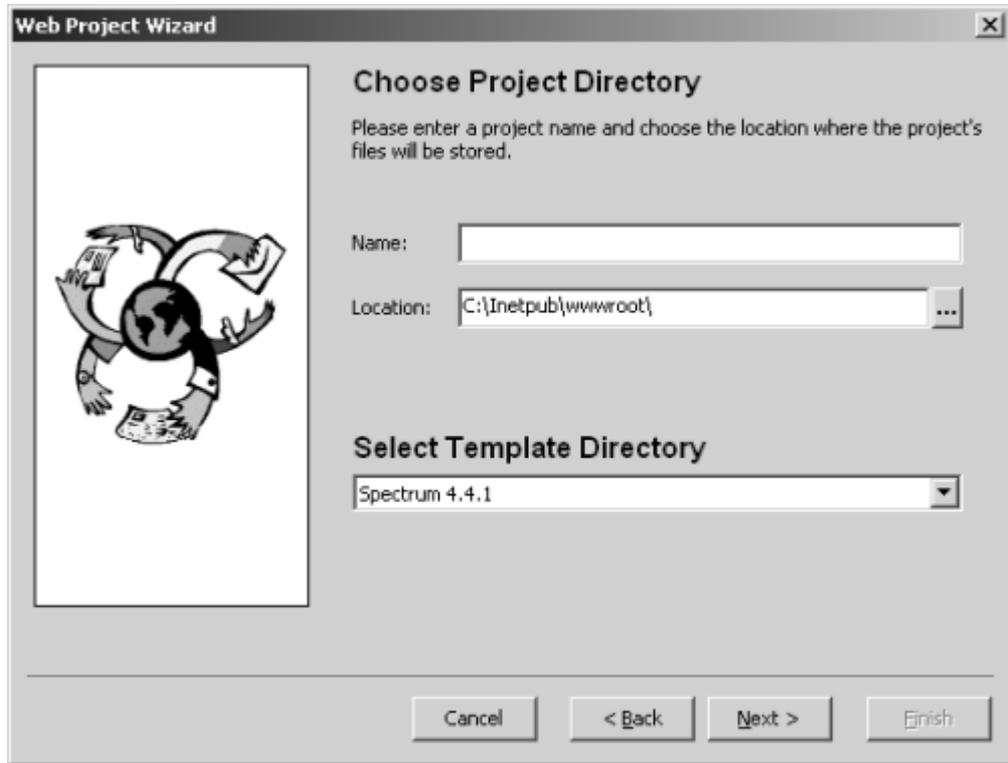
- 3 Select Spectrum Web Project.

- 4 Click Open.
The Web Project wizard is displayed:



Web Project Wizard

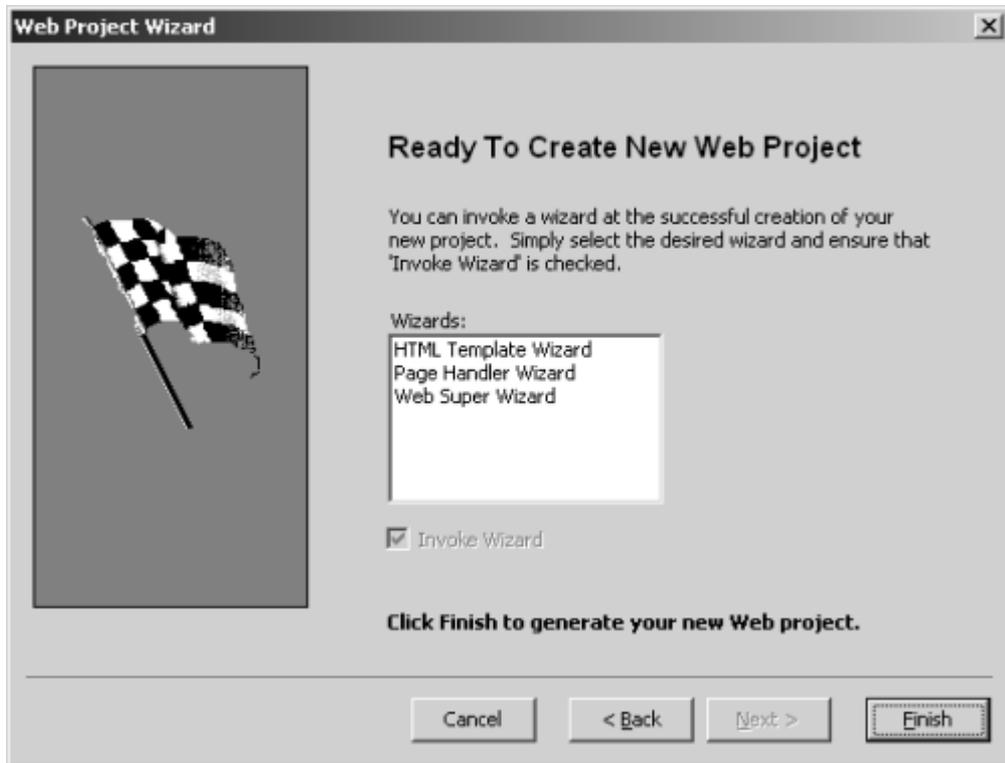
- 5 Click Next.
The Choose Project Directory window is displayed:



Web Project Wizard — Choose Project Directory Window

- 6 Type the name of the web project.
- 7 Specify a directory where the project files will be stored.
We recommend that you use C:\inetpub\wwwroot as the directory. If you do not use this directory, you must add a virtual directory in IIS when you deploy the application. For information, see **Deploying Your Application**, page 155.
- 8 Select the template directory.
This option allows you to create different versions of the templates for the same web project.

- 9 Click Next.
The Ready to Create New Web Project window is displayed:



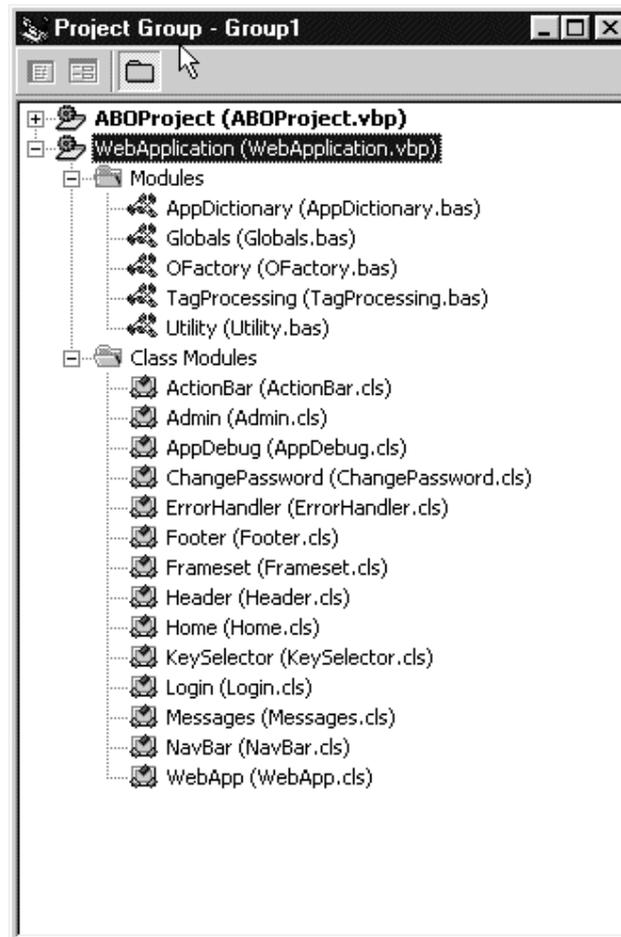
Web Project Wizard — Ready to Create New Project Window

At this point, you can:

- Click Finish to generate the web project.
- Select the Web Super wizard to generate the HTML template and page handler framework components.
- Select an individual wizard to generate the corresponding framework component.

After selecting a wizard, ensure that the Invoke Wizard check box is marked.

- Click Finish.
The web project is generated and added to the project group:



New Web Project in Visual Basic Project Explorer

For information about the supplied framework components, see **Supplied Framework Components**, page 63.

SUPPLIED FRAMEWORK COMPONENTS

Construct Spectrum includes a set of generic, customizable framework components that provide the basic structure and appearance of a generated web application. This chapter describes the supplied framework components.

The following topics are covered:

- **Introduction**, page 64
- **Active Server Page (ASP) Script and Files**, page 65
- **BAS (.bas) Files**, page 66
- **Cascading Style Sheet (.css) File**, page 67
- **HTML Templates and Page Handlers**, page 68
- **JavaScript (.js) Files**, page 73

Introduction

When you create a Construct Spectrum web project, several framework components are added to the application directory. These components include support files, HTML templates for standard page components, page handlers to process the HTML templates, ASP script, graphics, code frames, JavaScript, and cascading style sheets.

The framework components provide a consistent look and feel to your application, as well as decrease development efforts. Changes to the framework components affect how all generated components look and perform.

Using the framework components, the Spectrum wizards, and an existing ActiveX business object (ABO), you can generate a functional web application. First generate the page handler, HTML template, and object factory entries for the ABO, then customize the generated components and supplied framework components to suit the purpose and design of your application.

The following sections describe the framework components and their functions.

Active Server Page (ASP) Script and Files

Construct Spectrum supplies an Active Server Page (ASP) script and associated files as framework components. This script activates the WebApp class, which starts the Construct Spectrum web application.

The following files are supplied:

File	Description
Global.asa	ASA file that detects session and application events. It detects when a user starts and ends a session. It can also detect when the web application starts for the first time and when it ends.
WebApp.asp	ASP file that accesses the application in Nonframes mode.
WebAppF.asp	ASP file that accesses the application in Frames mode.
WebAppFS.asp	ASP file that requests the frame set describing the layout of web pages in Frames mode.
WebApp.cls	Module that is the public creatable component in the application, providing a starting point for requests to the application. This module is called by an ASP file when a web browser requests the page. It retrieves session and application information from the web server and passes the information to the application associated with the ASP script. This module contains separate routines for Frames and Nonframes modes.

Note: WebApp.cls runs on the Microsoft Transaction Server only.

Note: When updating the object factory, these files are also updated.

BAS (.bas) Files

Construct Spectrum supplies several BAS (.bas) files as framework components. BAS files provide global functions for an application. This reduces future maintenance efforts, as global changes can be applied by editing a single file.

The following BAS files are supplied:

File	Description
AppDictionary.bas	Dictionary file containing translations for terms used in HTML pages. Use this file to provide alternate field names. For example, the KeySelector.cls page handler calls this file and checks for names of fields used in the template. Since the key selector uses these names to determine search criteria, you can specify meaningful names for the descriptor fields in files accessed by the application.
OFactory.bas	Object factory that co-ordinates the creation of objects when running the application. It also contains security routines to validate users and evaluate security tags in HTML templates. You can update the object factory using the Object Factory wizard. For information, see Updating and Customizing the Object Factory , page 137.
Globals.bas	Globals file containing settings and routines used throughout your application. To enable security for the Login page, modify this file. For information, see Customizing the Globals.bas File , page 152.
TagProcessing.bas	Tag parser containing generic tag processing used by the application. It also contains a routine you can customize to create new tags. Use this file to write customizations for the HTML replacement tags used by the application. For information, see Using HTML Replacement Tags , page 121.
Utility.bas	File containing utility and help routines.

Cascading Style Sheet (.css) File

Construct Spectrum supplies the Styles.css cascading style sheet as a framework component. A cascading style sheet defines a set of styles that override the web browser's standard methods for rendering HTML. It defines the look of elements on web pages, which gives your pages a unique and consistent design. A style sheet allows you to define the attributes of any tag. For example, you can set the font, line spacing, justification, and border properties.

To apply the style sheet to a page, add a <LINK> tag between <HEAD> and </HEAD> in the HTML document heading.

In Construct Spectrum-generated web applications, the style sheet links are included in the Layout.htm HTML template, which determines the layout and appearance of all web pages in the application. For example:

```
<LINK REL=stylesheet HREF="support\styles.css" TYPE="text/css">
```

Because all the style information is stored in a single file, it is easier to maintain and saves space on the web server.

HTML Templates and Page Handlers

Construct Spectrum supplies several HTML templates as framework components. Use these templates to create components that are common to all web pages, such as the navigation bar. Most of the supplied HTML templates also have an associated page handler, which processes HTML replacement tags by retrieving live content or another HTML template.

To view many of the following framework components, open the demo application or see **Features of the Demo Application**, page 35.

The following HTML templates and associated page handlers are supplied:

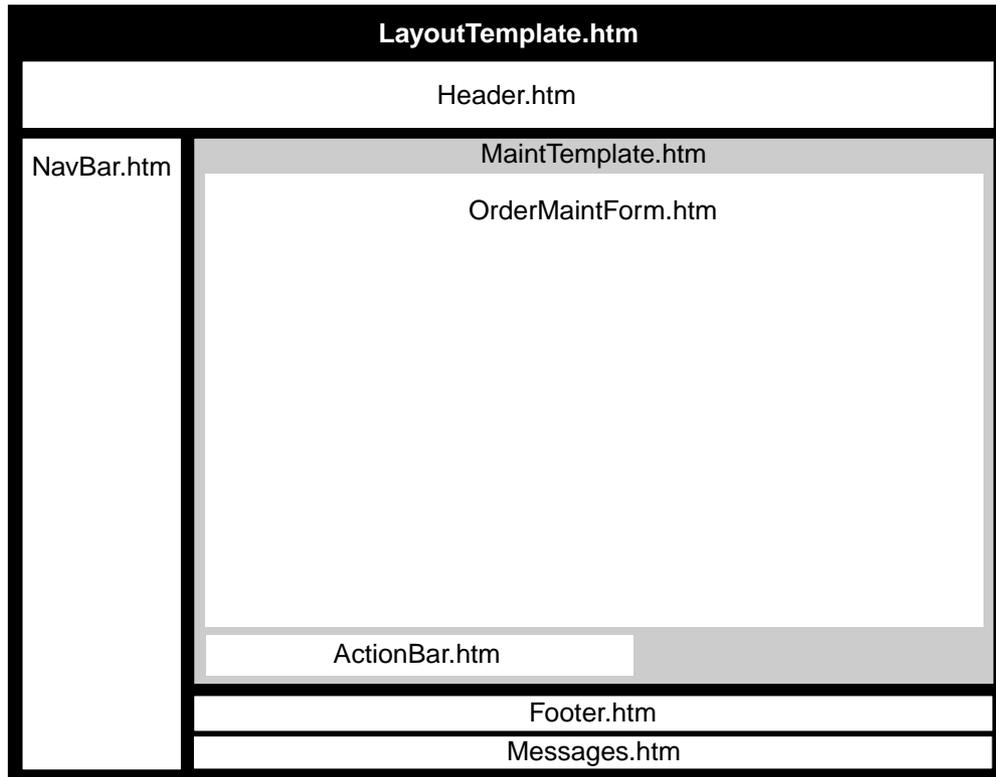
File	Description
ActionBar.cls ActionBar.htm	Action bar showing methods available for the business object on a maintenance page. The action bar displays the methods as buttons, such as Read, Next, Clear, Add, Update, and Delete.
Admin.cls Admin.htm	Administration page, which is standard to all Construct Spectrum-generated web applications. This page allows you to change the dispatch service used by the web server to access the mainframe. By default, the navigation bar contains a link to this page.
AppDebug.cls AppDebug.htm	Debug page, which is standard to all Construct Spectrum-generated web applications. This page displays information about user and application settings cached on the web server. By default, the navigation bar contains a link to this page.
BestViewed.htm	Page displayed when the web browser does not meet the minimum requirements. For information, see Supported Web Browsers , page 22.
BrowseTemplate.htm	HTML template that determines the layout of browse pages in the web application.
Calendar.htm	Calendar pop-up containing date information. Users can use drop-down lists to select a month and year and then click the day on the calendar to select the date.
ChangePassword.cls ChangePassword.htm	Change Password page, which is standard to all Construct Spectrum-generated web applications. This page allows users to change the passwords used to login to the application. By default, the navigation bar contains a link to the Change Password page.

File	Description (continued)
ErrorHandler.cls ErrorHandler.htm	Files that display the standard response page when an invalid or unknown page is requested.
Footer.cls Footer.htm	Footer, which is a standard component of every web page. For example, you can customize Footer.htm to display links to other pages or your corporate logo.
Frameset.cls Frameset.htm	Files responsible for the appearance and performance of web pages operating in Frames mode.
Header.cls Header.htm	Header modules responsible for displaying content that is standard on every page in your application, as well as custom content specified in the generated HTML template for the page.
Home.cls Home.htm	Home page, which is standard to all Construct Spectrum-generated web applications. This page provides users with a starting point from which to access other pages.
KeySelector.cls KeySelector.htm	Key selector components used by all browse pages to display generic key selection options. The page handler queries the browse at runtime and uses the information gathered to display the logical key selection and key components.
LayoutTemplate.htm	HTML template that determines the layout of web pages operating in Nonframes mode.
Login.cls Login.htm	Login page, which is standard to all Construct Spectrum-generated web applications. This page allows users to access the web application's business objects. The Login.cls page handler calls the object factory to validate the user ID and password.
Logout.htm	Logout page, which is standard to all Construct Spectrum-generated web applications. It is displayed when the user logs out of the application.
MaintTemplate.htm	HTML template that determines the layout for maintenance pages in the application.

File	Description (continued)
Messages.cls Messages.htm	Messages area, which is a standard component of every web page. Located above the footer, it displays status and error messages.
NavBar.cls NavBar.htm	<p>Navigation bar, which is standard to all Construct Spectrum-generated web applications.</p> <p>Displayed along the left edge of each web page, the navigation bar presents links to other web pages in the application. By default, it displays the following standard options: Login/Logout, Change Password, Admin, Home, Debug, and Frames.</p> <p>When the user logs in, the NavBar.cls page handler builds the navigation bar by querying the object factory for information about the user's permissions to access business objects. Only the business objects and functions available for that user are displayed. NavBar.cls also builds the flyout menus.</p>

Examples of HTML Templates

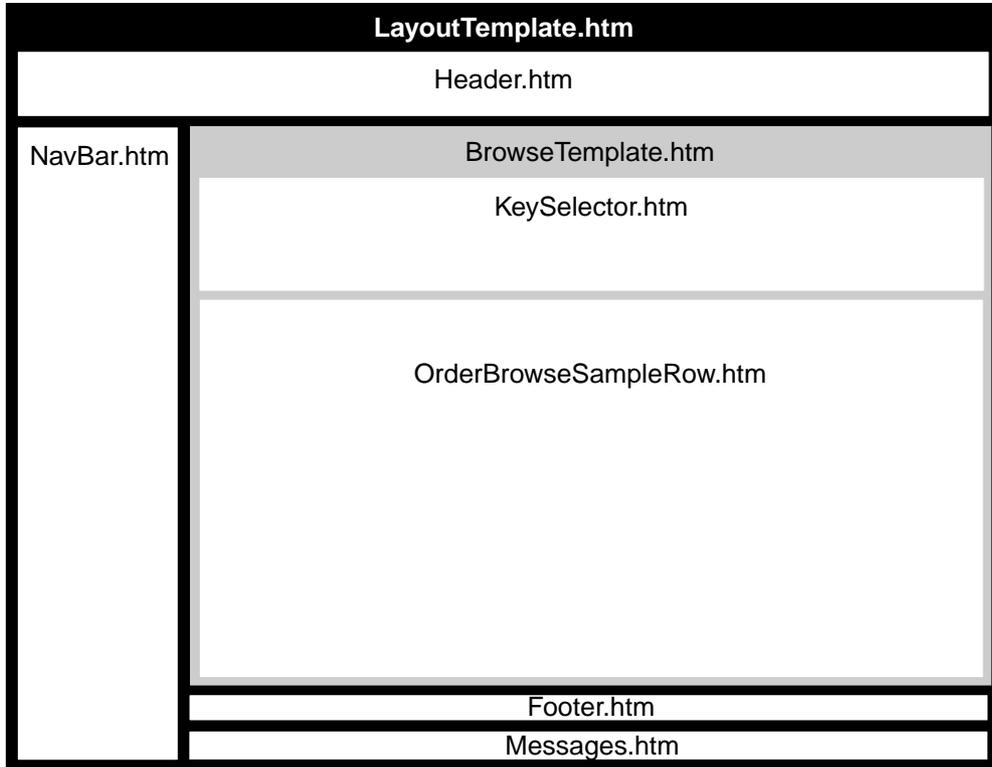
The following example shows how the generated HTML templates and the supplied HTML templates work together to build a maintenance page:



HTML Templates Used to Build a Maintenance Page

In this example, the only generated HTML template is OrderMaintForm.htm. Construct Spectrum supplies all other HTML templates, along with their associated page handlers, as framework components.

The following example shows how the generated HTML templates and the supplied HTML templates work together to build a browse page:



HTML Templates Used to Build a Browse Page

In this example, the only generated HTML template is `OrderBrowseSampleRow.htm`. Construct Spectrum supplies all other HTML templates, along with their associated page handlers, as framework components.

JavaScript (.js) Files

Construct Spectrum supplies several JavaScript files as framework components. JavaScript files perform standard functionality.

The following JavaScript files are supplied:

File	Description
Browse.js	<p>Contains scripts that provide common functions to browse pages, such as:</p> <ul style="list-style-type: none"> • Convert a logical key name into a legal identifier by removing all hyphens and blanks from a name string • Display the search key field values for a given key • Set the visibility for a section • Show the fields for the currently selected logical key • Change an inner HTML value <p>For the Internet Explorer browser, it also provides the following functions:</p> <ul style="list-style-type: none"> • Copy text box values to the master form whenever the values change • Save the visibility setting for the key selector
Common.js	<p>Contains scripts that provide common functions to all web pages, such as:</p> <ul style="list-style-type: none"> • Get the key value for a cookie • Set the key value for a cookie • Enable or disable the elements on a form • Refresh the navigation bar in the navbar frame • Set the action field and submit a form • Submit a URL based on the contents of a form element • Submit a URL • Bold graphics • Fade graphics • Hide or show a section (collapsible sections)
Maint.js	<p>Contains scripts that provide common functions to maintenance pages, such as:</p> <ul style="list-style-type: none"> • Display or collapse a section using the “+” and “-” functionality • Highlight the errors on a form
Menu.js	<p>Contains scripts that build the flyout (pull-down) menus.</p>

File	Description (continued)
Menu.xsl	Contains the xsl used to transform flyout menus.
MenuCodeframe.xsl	Generates the xml when a flyout menu is selected by the Object Factory wizard.

For information about the supported versions of Microsoft Internet Explorer and Netscape Navigator, see **Supported Web Browsers**, page 22.

CREATING AND CUSTOMIZING A PAGE HANDLER

This chapter describes how to use the Page Handler wizard to generate page handlers for your web application. It also describes how to use the supplied user exits to customize your page handlers.

The following topics are covered:

- **Generating a Page Handler**, page 76
- **Customizing a Page Handler**, page 84

Generating a Page Handler

A page handler is a Visual Basic class within your web project. Page handlers respond to user requests, return HTML to the browser, and access ABOs to invoke business object properties and methods. They also process replacement tags in the HTML templates. Typically, you generate a page handler for each ABO in your application.

- To generate a page handler using the Page Handler wizard:
 - ❑ **Step 1: Invoke the Wizard**, page 77.
 - ❑ **Step 2: Select an ABO**, page 78.
 - ❑ **Step 3: Confirm Details about the ABO**, page 80.
 - ❑ **Step 4: Configure the Page Handler**, page 81.
 - ❑ **Step 5: Generate the Page Handler**, page 83.

The following sections describe these steps in detail.

Step 1: Invoke the Wizard

The Page Handler wizard is available as part of a Construct Spectrum Add-In to Visual Basic.

- To invoke the Page Handler wizard from Visual Basic:
 - 1 Open the project group for your application.
 - 2 Select **Wizards > Page Handler** from the Spectrum menu.
The Page Handler wizard is displayed:



Page Handler Wizard

For information about the Configuration editor or Spectrum Cache viewer (represented by the icons in the lower left of the window), see **Features of the Wizards**, page 49, *Construct Spectrum SDK Reference*.

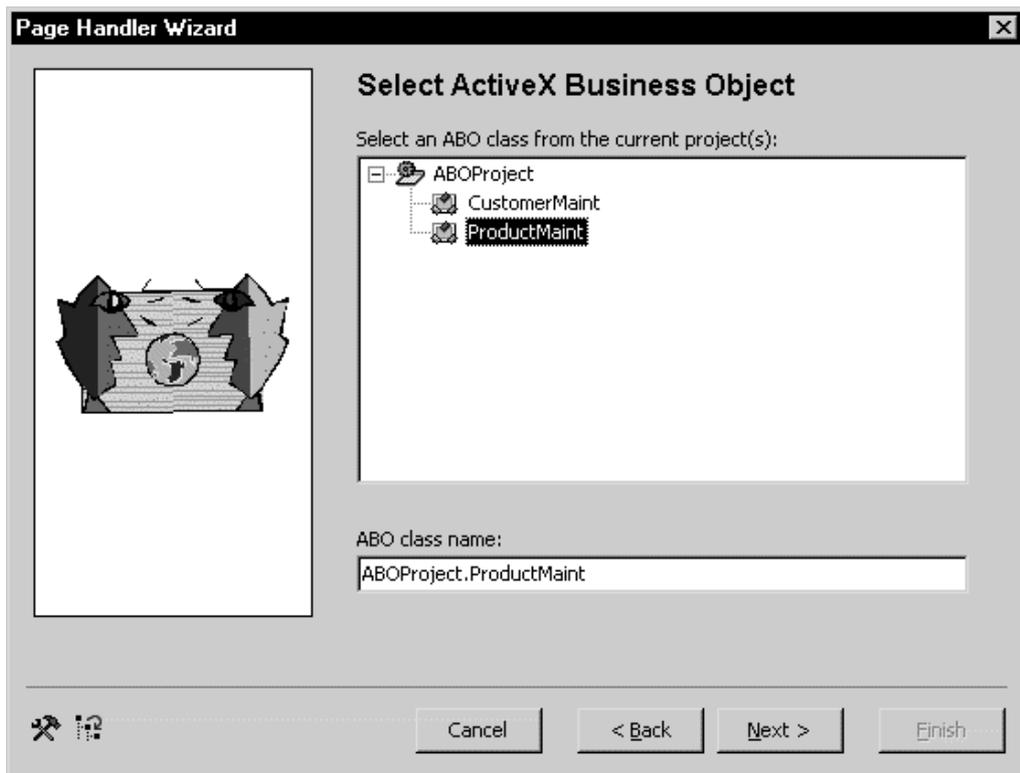
Step 2: Select an ABO

The ABO (ActiveX business object) represents a Natural subprogram proxy defined on the mainframe to access database fields. The ABO encapsulates the subprogram, which specifies a Predict view and selected fields, and makes the fields available to your web application as properties of the ABO.

➤ To select an ABO:

1 Click Next.

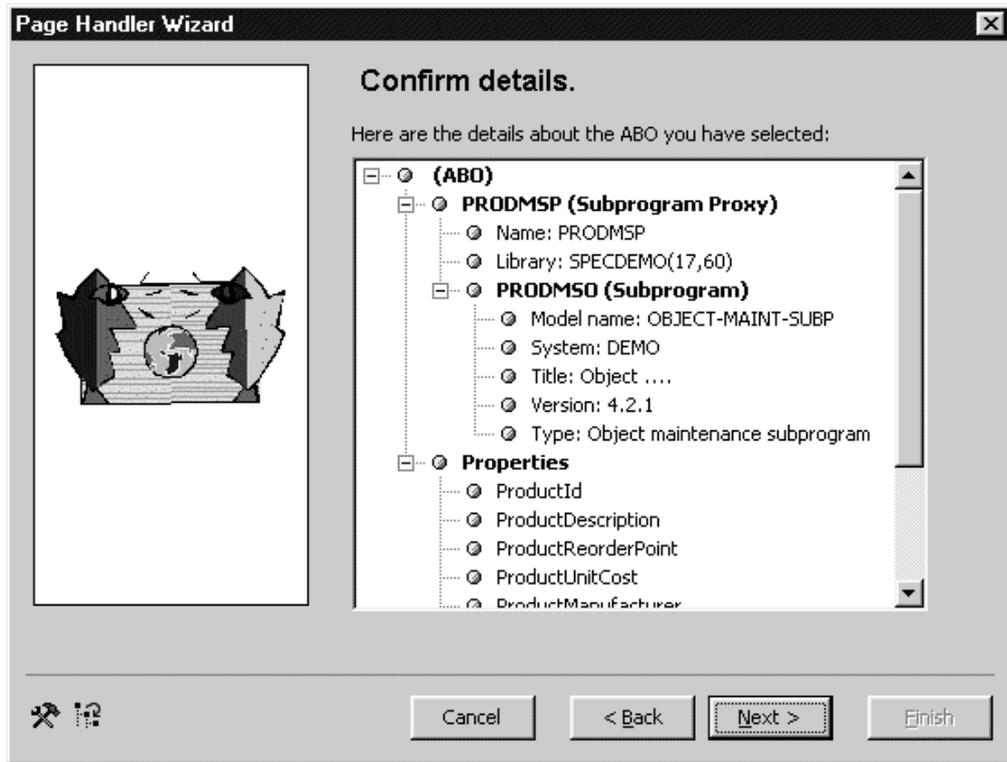
The Select ActiveX Business Object window is displayed:



Page Handler Wizard — Select ActiveX Business Object Window

2 Select an ABO from the ABO project.

- 3 Click Next.
A summary of the ABO is displayed:



Page Handler Wizard — Confirm Details Window

Step 3: Confirm Details about the ABO

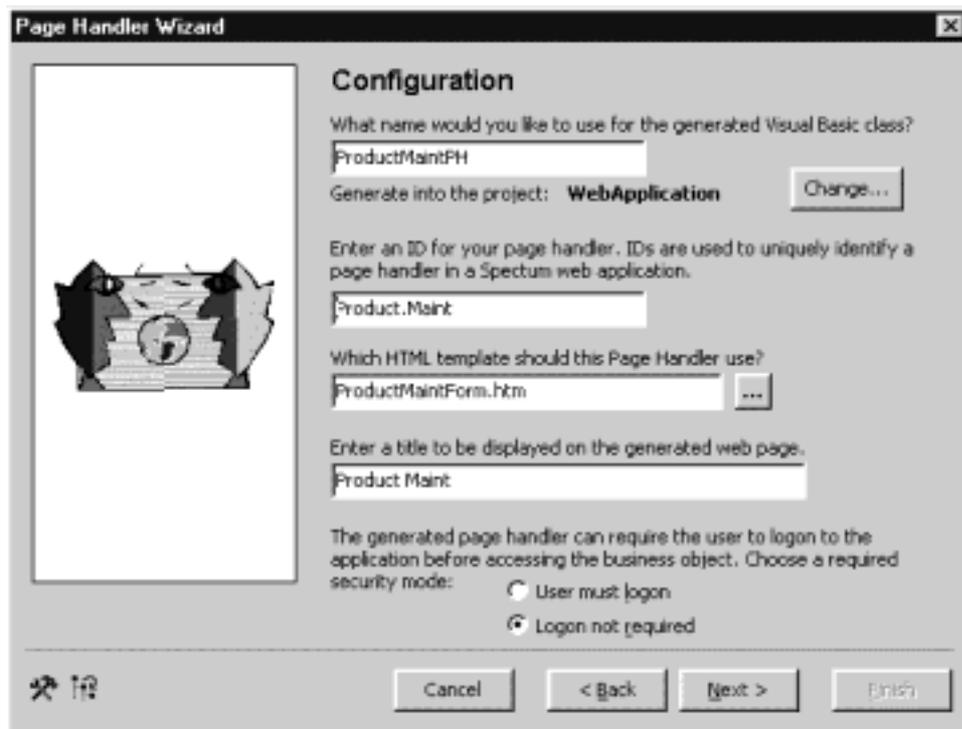
The ABO summary shows details about the subprogram proxy and Natural subprogram the ABO encapsulates, as well as the properties associated with the ABO.

➤ To confirm details about the ABO:

1 Click Next.

The wizard reads the ABO specification to confirm that the functionality is supported. (If not, a message is displayed asking you to select a different ABO class.) It also reads the field names and properties exposed by the ABO.

After selecting an ABO and clicking Next, the configuration options are displayed:



The screenshot shows the 'Page Handler Wizard' window with the 'Configuration' tab selected. The window is titled 'Page Handler Wizard' and has a close button in the top right corner. On the left side, there is a large empty rectangular area with a small icon of a page handler in the center. The right side contains the following configuration options:

- Configuration**
- What name would you like to use for the generated Visual Basic class?
ProductMaintPH
- Generate into the project: **WebApplication** (Change...)
- Enter an ID for your page handler. IDs are used to uniquely identify a page handler in a Spectrum web application.
Product.Maint
- Which HTML template should this Page Handler use?
ProductMaintForm.htm (...)
- Enter a title to be displayed on the generated web page.
Product Maint
- The generated page handler can require the user to logon to the application before accessing the business object. Choose a required security mode:
 - User must logon
 - Logon not required

At the bottom of the window, there are four buttons: Cancel, < Back, Next >, and Finish.

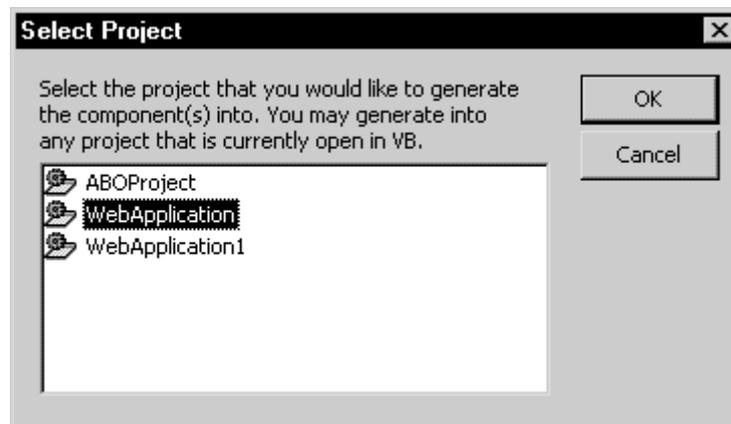
Page Handler Wizard — Configuration Window

Step 4: Configure the Page Handler

- To configure the page handler:
- 1 Verify the default name for the generated Visual Basic class.
The default class name is based on the object (for example, Product) and action (for example, Maintenance) represented by the ABO, with the addition of “PH” to indicate that the module is a page handler. You can change the file name under which the page handler is saved by typing a new name in the text box.

Tip: To change how default names are derived for page handlers, use the Configuration editor. For information, see **Invoke the Configuration Editor**, page 50, *Construct Spectrum SDK Reference*.

- 2 Verify the default name for the web project.
The web project name is displayed under the default class name. To save the file to another project, click Change. The Select Project window is displayed:



Select Project Window

This window lists the projects in the current ABO project group. You can select another project from the group and click OK.

- 3 Verify the default ID for the page handler.
This ID is used internally (for example, to provide links between pages).
- 4 Verify the default HTML template.
To select another template from a list of available templates, click the browse button. For information about HTML templates, see **HTML Templates and Page Handlers**, page 68.

- 5 Specify the level of security you want to apply to the business object:
 - If you want users to logon to the application before accessing this page, select User must logon.
 - If you want users to access this page without being logged onto the application, select Logon not required.
For information about using the Logon functionality, see **Securing Your Application**, page 149.
- 6 Click Next.
The generation options are displayed:



Page Handler Wizard — Ready to Generate Window

Step 5: Generate the Page Handler

➤ To generate the page handler:

1 Click Generate.

This initiates a test generate, which allows you to preview the code before updating your Visual Basic project. It also produces a generation report.

If you have a code comparison utility installed and configured for use with Construct Spectrum, you can also compare the new code with code from an earlier generation of the module. For information about using a code comparison utility, see **Use Reports with a Code Comparison Tool**, page 67, *Construct Spectrum SDK Reference*. For information about the generation report, see **Generating and Reviewing Reports**, page 63, *Construct Spectrum SDK Reference*.

Note: If you want to generate another page handler when generation is complete, select Yes.

2 Click Finish.

Construct Spectrum generates the code, updates the Visual Basic project, and closes the wizard. When generation is completed, a confirmation message is displayed to inform you of the success or failure of the operation.

Tip: If there are problems with generation, click Generate to view the generation report.

Customizing a Page Handler

You can customize page handlers so that they perform tasks specific to your web application. For example, you can modify the security logic for your application or add your own HTML tags. For information about customizing tags, see **Using HTML Replacement Tags**, page 121.

To customize a page handler, do one of the following:

- Add custom code in implied user exits
- Add custom code in supplied user exits
- Modify the generated code and then protect it when regenerating

The following sections describe each of these customization options.

Implied User Exits

You can use implied user exits to add custom code to any function or subroutine. Implied user exits act as placeholders for hand-coded user exits during generation, ensuring that they are placed properly in the source code. For information, see **Implied User Exits**, page 58, *Construct Spectrum SDK Reference*.

Supplied User Exits

The following sections describe the user exits supplied for page handlers. Unless otherwise noted, user exits apply to page handlers for both maintenance and browse functions.

Content.CustomContentIDs

Use this exit to extend the standard template content and process new templates based on a content ID supplied to the program. The exit is located in the `ICSTPageHandler_Content` function and is only available for maintenance page handlers.

Example of user exit code

The following example uses a template called `OrderMaintForm2.htm` whenever the ContentID `Form2` is specified:

```
'<cst:EXIT Name=Content.CustomContentID>  
Case "Form2"  
    ICSTPageHandler_Content = ParseTemplate("OrderMaintForm2.htm")  
'</cst:EXIT'
```

BDT.Overrides

Use this exit to change the BDT names for specific fields or change how an ABO logical format is translated into a BDT name. The exit is located in the ICSTPageHandler_Initialize subroutine.

Example of user exit code

The following example uses the Currency BDT for the OrderAmount field and the Alpha BDT for the OrderWarehouseID field. It also specifies that the Phone logical format uses a Numeric BDT (the default is to use a BDT with the same name as the logical format):

```
'<cst:EXIT Name="BDT.Overrides">
.BDT("OrderAmount") = "Currency"
.BDT("OrderWarehouseID") = "Alpha"
.LogicalFormatBDT("Phone") = "Numeric"
'</cst:EXIT>
```

ParseTemplate.CustomTags

Use this exit to support new or customized tags on the HTML template supported by this page handler. The exit is located in the ParseTemplate function.

Example of user exit code

The following example creates two new tags for use on any supported template. The first tag, DUE_DATE, calculates a value based on a field in the ABO:

```
'<cst:EXIT Name="ParseTemplate.CustomTags">
Case "DUE_DATE" ' This value is today's date plus 30 days.
  If Not (m_ABOInterface.GetField("OrderDate") = "") Then
    tag.Contents = Format(DateAdd("d", 30, _
      _ABOInterface.Field("OrderDate")), _
      "dd-mmm-yyyy")
    breplaced = True
  End If
Case "NOTES"
  tag.Contents = m_Notes
  breplaced = True
'</cst:EXIT>
```

For more information, see **Using HTML Replacement Tags**, page 121.

PerformAction.OtherResets

Use this exit to reset custom values when the user retrieves a new record. The exit is located in the PerformAction function and is only available for maintenance page handlers.

Example of user exit code

In the following example, the m_Notes variable is reset to an empty string:

```
'<cst:EXIT Name="PerformAction.OtherResets">
m_Notes = ""
'</cst:EXIT>
```

PerformAction.CustomUpdateActions

Use this exit to add customized actions that update the ABO based on the Action parameter. The exit is located in the PerformAction function and is only available for maintenance page handlers.

Example of user exit code

The following example shows how a custom action, called RecalcDates, updates the ABO using the data on the HTML template:

```
'<cst:EXIT Name="PerformAction.CustomUpdateActions">
Case "RecalcDates"
    If Not UpdateData(True) Then Exit Sub
'</cst:EXIT>
```

PerformAction.UpdateForeignKeys

Use this exit if your maintenance object supports foreign key lookups. The HTML template wizard generates the HTML elements to start the lookup, but the code in this exit must update the ABO with the fields returned by the foreign key browse. The exit is located in the PerformAction function and is only available for maintenance page handlers.

Example of user exit code

The following example shows how the OrderWarehouseID field is updated when the Warehouse.Browse page handler browses by a foreign key:

```
'<cst:EXIT Name="PerformAction.UpdateForeignKeys">
Case "Warehouse.Browse"
    m_ABOInterface.SetField abo.ABOInterface.GetField("WarehouseID", _
                                                m_RequestData.Request("Row")), "OrderWarehouseID"
'</cst:EXIT>
```

PerformAction.ClientValidations

Use this exit to add custom validations for data on the client. The exit is located in the PerformAction function and is only available for maintenance page handlers.

Example of user exit code

The following example shows how to add a validation that ensures an order date is within an acceptable range (in this case, greater than today). The example checks to see if a custom action, called RecalcDates, is executed. As the RecalcDates action updates the field, no error checking is performed:

```
'<cst:EXIT Name="PerformAction.ClientValidations">
  If m_ABOInterface.Error("OrderDate").ErrorMsg = "" Then
    If saction <> "RecalcDates" Then
      If CDate(m_ABOInterface.GetField("OrderDate")) < Now() Then
        m_ABOInterface.AddError "OrderDate", _
          "OrderDate must be later than today", _
          "OrderDate",
m_ABOInterface.GetField("OrderDate")
        AddErrorMessages m_RequestData, m_ABOInterface
      Exit Sub
    End If
  End If
End If
'</cst:EXIT>
```

PerformAction.CustomActions

Use this exit to write code for custom maintenance actions. The exit is located in the PerformAction function and is only available for maintenance page handlers.

Example of user exit code

The following example adds a custom action to recalculate a date field. A button is placed on the maintenance form to trigger this action:

```
'<cst:EXIT Name="PerformAction.CustomActions">
Case "RecalcDates"
  ' OrderDate should be two days after the current date.
  m_ABOInterface.SetField DateAdd("d", 2, Now()), "OrderDate"
'</cst:EXIT>
```

RetrieveFromSession.CustomState and StoreToSession.CustomState

Use these exits to allow the page handler to cache additional data in the session object. The exits are located in the RetrieveFromSession and StoreToSession subroutines.

Example of user exit code

The following example shows how to store the value of a module-level variable in the session object:

```
'<cst:EXIT Name="StoreToSession.CustomState">  
sn.Value("Notes") = m_Notes  
'</cst:EXIT>
```

Example of user exit code

The following example shows how to retrieve the value of a module-level variable from the session object:

```
'<cst:EXIT Name="RetrieveFromSession.CustomState">  
m_Notes = sn.Value("Notes")  
'</cst:EXIT>
```

UpdateData.CustomUpdates

Use this exit to update custom data that does not exist in the ABO. The exit is located in the UpdateData function and is only available for maintenance page handlers.

Example of user exit code

The following example shows how to use an item in the Request object to update a module-level variable (previous examples show how this field is used):

```
'<cst:EXIT Name="UpdateData.CustomUpdates">  
m_Notes = m_RequestData.Request("Notes")  
'</cst:EXIT>
```

Process.CustomActions

Use this exit to code custom browse actions. The exit is located in the Process function and is only available for browse page handlers.

Example of user exit code

The following example shows how to add an action, called BROWSEALL, which retrieves all records in the database and returns the HTML content to the user:

```
'<cst:EXIT Name="Process.CustomActions">
Case "BROWSEALL"
  Do While Not bo.EndOfData
    PerformBrowse True
  Loop
'</cst:EXIT>
```

Modifying and Protecting Generated Code

You can modify the generated code and then protect the customizations during regeneration by enclosing the customized portion using a `cst:PRESERVE` tag. For information, see **Preserve Customizations to Generated Code**, page 58, *Construct Spectrum SDK Reference*.

CREATING AND CUSTOMIZING AN HTML TEMPLATE

This chapter describes how to use the HTML Template wizard to create HTML templates for your application. It also describes how to customize your HTML templates before generation.

Each HTML template presents a web page, or component of a web page, that the associated page handler interprets and assembles at runtime. When building maintenance and browse pages, use the supplied HTML templates to provide standard functionality and use the HTML Template wizard to generate business-specific HTML templates.

Note: For information about the supplied HTML templates, see **HTML Templates and Page Handlers**, page 68.

The following topics are covered:

- **Introduction**, page 92
- **Using the HTML Template Wizard**, page 94
- **Customizing HTML Before Generation**, page 103

Introduction

An HTML template is a file that defines a web page or component of a web page. Construct Spectrum provides the HTML Template wizard to generate HTML templates for your web applications. The wizard uses information stored in an ABO to select the content and layout of a maintenance or browse page.

In addition to the HTML code, generated HTML templates use:

- JavaScript to perform some processing
- Links to cascading style sheets that provide a consistent look to pages throughout the application

Supplied HTML Templates

In addition to the HTML templates generated using the HTML Template wizard, your web pages use HTML templates supplied with Construct Spectrum. These templates provide standard page components, such as the navigation bar, header, footer, and message area. They also provide standard layouts for web pages, depending on whether the page supports maintenance or browse functionality.

For a list of all supplied templates and information about how the supplied and generated HTML templates work together to create a web page, see **HTML Templates and Page Handlers**, page 68.

HTML Replacement Tags

Wizard-generated HTML templates make use of HTML replacement tags, which act as placeholders for other templates or active content retrieved from the mainframe database. At runtime, the page handler assembles the HTML templates (beginning with the designated starting template), parses the tags, and resolves references to sub-templates and active content. Templates are processed until all replacement tags are resolved. The resulting HTML is sent to the browser as an HTTP response.

Customizing HTML Templates

Before Generating the Template

While using the HTML Template wizard to define an HTML template, you can customize the template before generation by modifying specifications for some HTML tags. For information, see **Customizing HTML Before Generation**, page 103.

After Generating the Template

After using the HTML Template wizard to generate a template, you can customize the HTML as desired. However, your changes are lost if you regenerate the template. To avoid this, rename your HTML template, generate a new template, and copy your customizations to the new template. If you have a code comparison utility installed and configured for use with Construct Spectrum, compare the regenerated HTML with the original HTML. For information, see **Features of the Wizards**, page 49, *Construct Spectrum SDK Reference*.

Creating Custom Replacement Tags

Another way to customize web pages is to create your own HTML replacement tags. Processing of custom tags is not affected when you regenerate an HTML template or page handler because it can be protected in a user exit in the page handler or ABO. Custom code can also be protected using the PROTECT tag.

For information about creating tags, see **Using HTML Replacement Tags**, page 121.

Using Supplied Framework Components

To perform customizations on web pages used throughout an application, modify the supplied HTML templates and page handlers, JavaScript, and cascading style sheets. This is the most effective way of maintaining consistency in appearance and processing in your application. For information, see **Supplied Framework Components**, page 63.

An alternative is to incorporate customizations into the wizards, which allows global changes to be generated into the template. To customize the wizards, modify the code frames supplied with Construct Spectrum.

Using the HTML Template Wizard

Use the HTML Template wizard to generate templates for each ABO in your application. For information about creating ABOs, see **Using ActiveX Business Objects**, page 85, *Construct Spectrum SDK Reference*.

- To generate an HTML template:
 - ❑ **Step 1: Invoke the Wizard**, page 95
 - ❑ **Step 2: Select an ABO**, page 96
 - ❑ **Step 3: Confirm ABO Details**, page 97
 - ❑ **Step 4: Configure the HTML Template**, page 98
 - ❑ **Step 5: Customize the HTML Template (Optional)**, page 100
 - ❑ **Step 6: Generate the HTML Template**, page 100

The following sections describe these steps in detail.

Step 1: Invoke the Wizard

The HTML Template wizard is part of a Construct Spectrum Add-In to Visual Basic.

- To invoke the HTML Template wizard:
 - 1 Open the project group for your application in Visual Basic.
 - 2 Select Wizards > HTML Template from the Spectrum menu.
The HTML Template wizard is displayed:



HTML Template Wizard

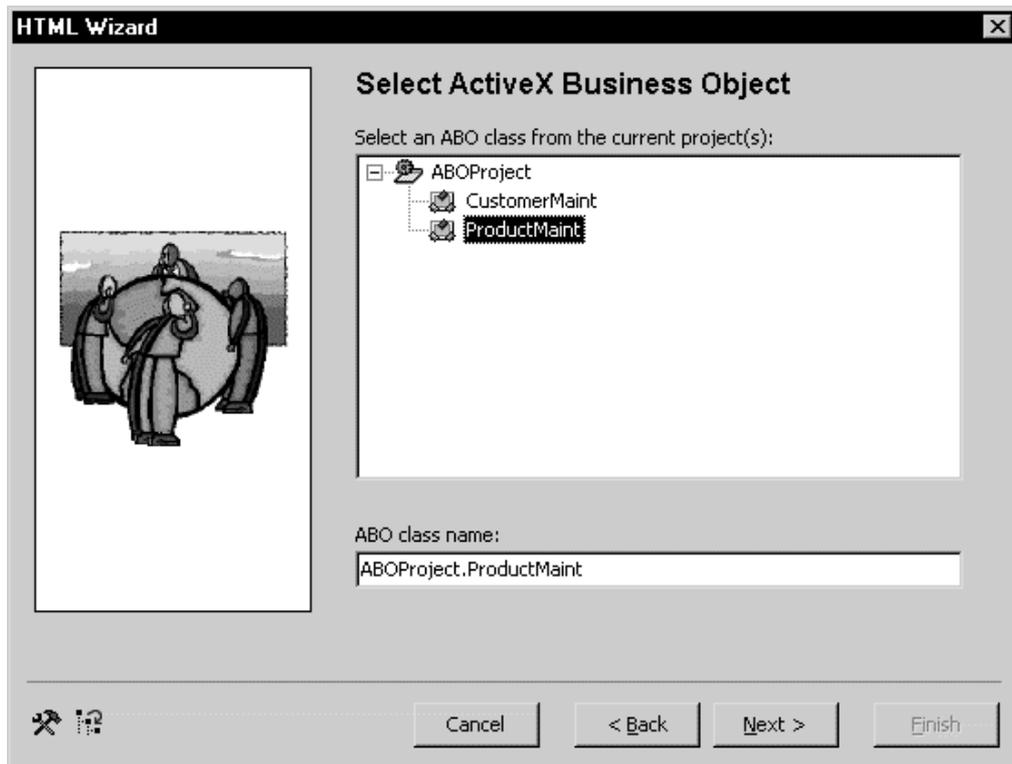
Note: For information about the Configuration editor or Spectrum Cache viewer (represented by the icons in the lower left of the window), see **Features of the Wizards**, page 49, *Construct Spectrum SDK Reference*.

- 3 Click Next.
The Select ActiveX Business Object window is displayed, showing the available ABOs in the current project.

The ActiveX business object is the Natural subprogram defined on the mainframe to access database fields. The Natural subprogram specifies a Predict view and selected fields. The ABO encapsulates the subprogram, making the fields available to your web application as properties of the ABO.

Step 2: Select an ABO

Select an ABO in the Select ActiveX Business Object window:

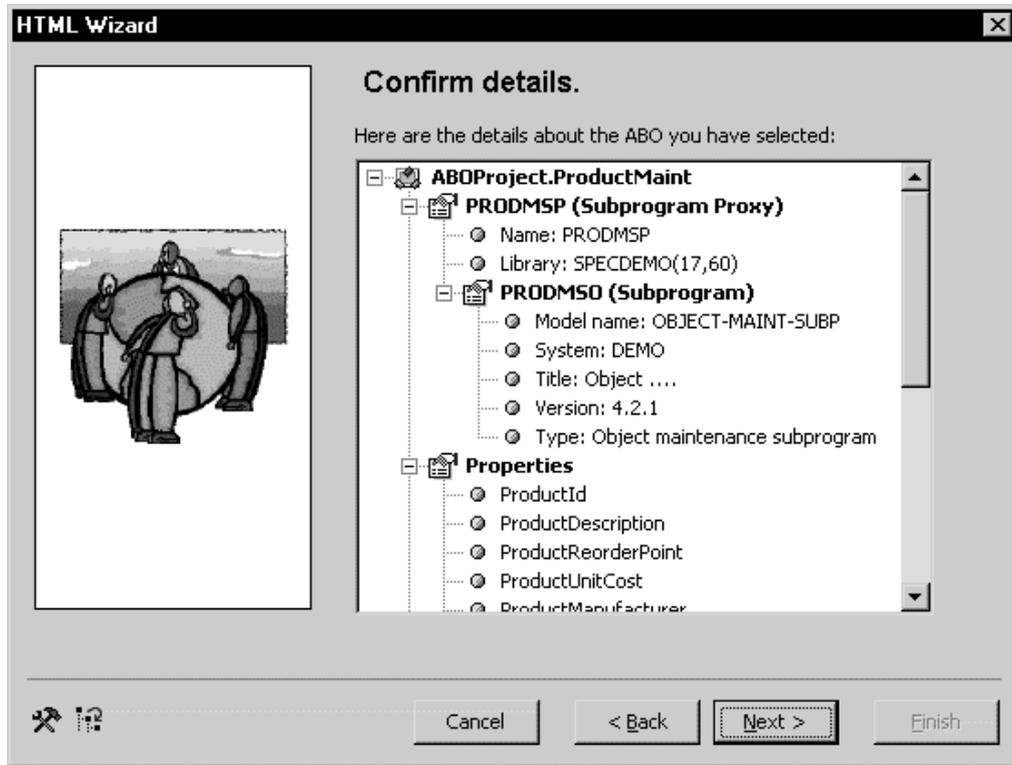


HTML Template Wizard — Select ActiveX Business Object Window

Click Next to display details about the ABO. The Confirm details window is displayed, showing the subprogram proxy and subprogram encapsulated by the ABO, as well as properties associated with it.

Step 3: Confirm ABO Details

The Confirm details window shows details about the selected ABO:



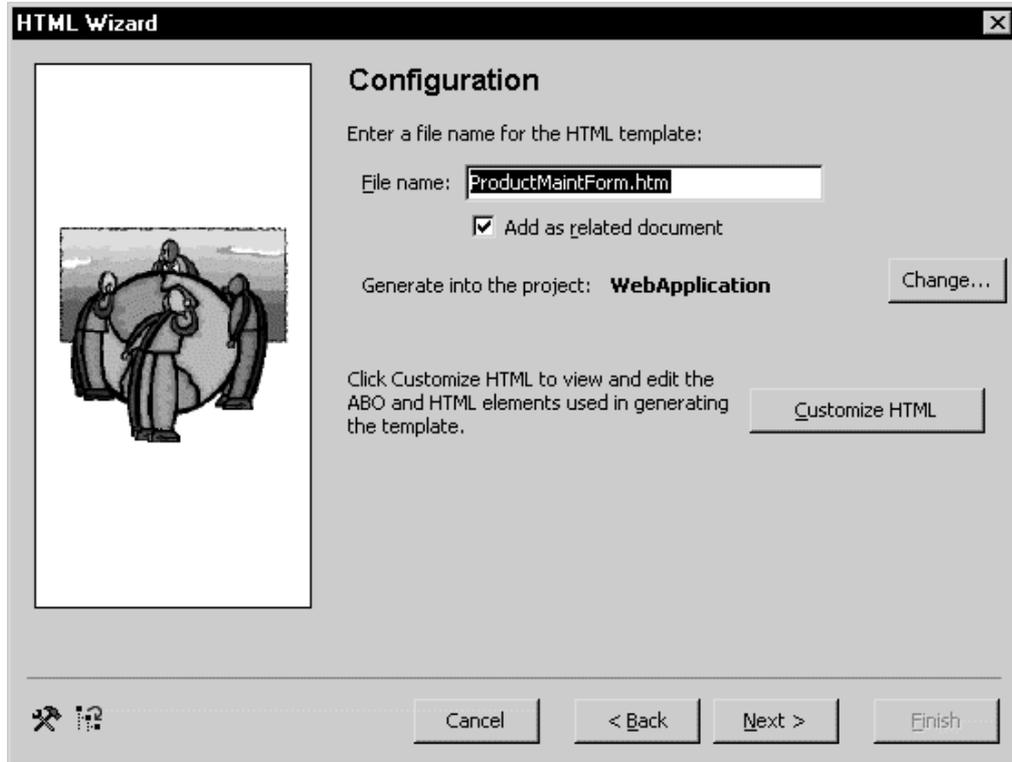
HTML Template Wizard — Confirm Details Window

Click Next to proceed. The wizard performs the following actions:

- Reads the ABO specification to confirm that the functionality it requires is supported. If the functionality is not supported, a message is displayed prompting you to select a different ABO class.
- Reads the field names and properties exposed by the ABO.

Step 4: Configure the HTML Template

After selecting an ABO and clicking Next, the configuration options are displayed:



HTML Template Wizard — Configuration Window

➤ To configure the HTML template:

- 1 Check the file name and change it, if necessary.

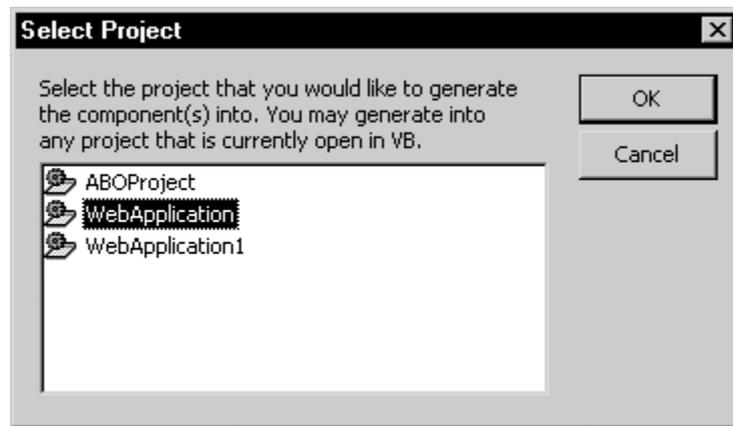
To change the file name under which the HTML template is saved, type the new name in File name. The HTML file is saved in the same directory as your web application.

Note: If you change the file name for the HTML template, be sure to change the page handler's reference to the name.

The default file name is based on the object (for example, Customer) and action (for example, Maintenance) represented by the ABO, with the addition of "Form" for a maintenance template or "SampleRow" for a browse template and the htm file extension.

Tip: You can use the Configuration editor to change how default names are derived for HTML templates. For information, **Invoke the Configuration Editor**, page 50, *Construct Spectrum SDK Reference*.

- 2 Ensure that the correct web project is selected and change it, if necessary. To save the file to another project, click Change to locate and select another project. The Select Project window is displayed:



Select Project Window

This window lists the projects in the current project group.

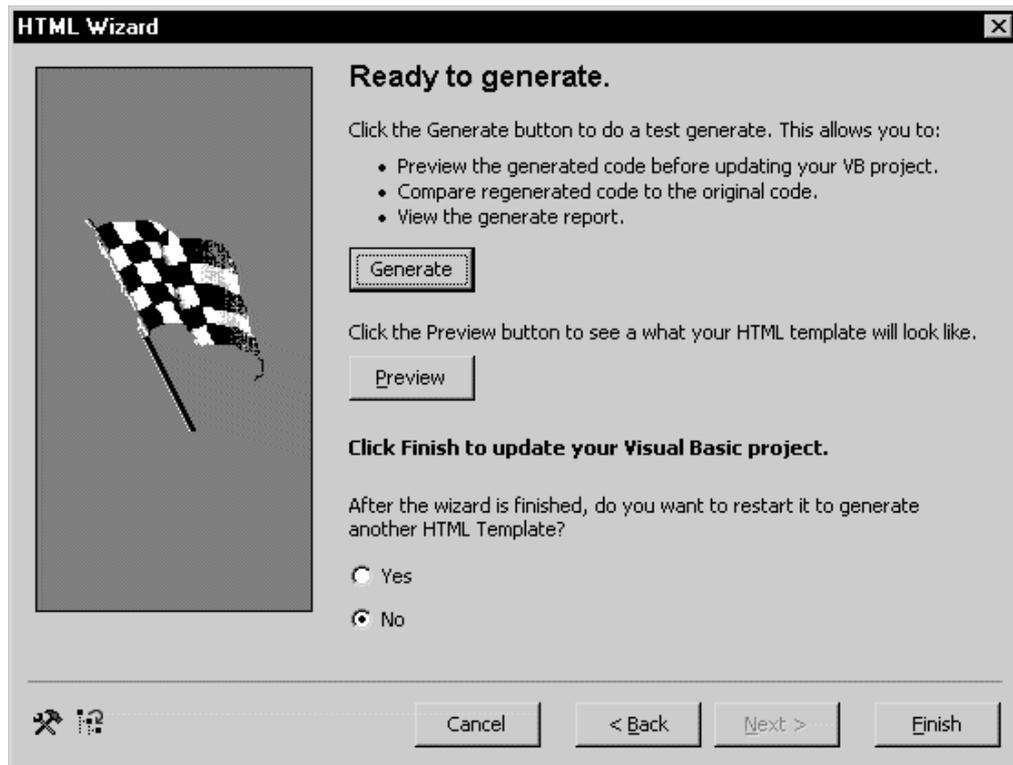
- 3 To have the HTML template appear in Project Explorer in the Related documents folder, select Add as related document. Showing generated HTML templates in Project Explorer makes it easier to keep track of the files in your project.

Tip: If you select the HTML file from the Project Explorer by double-clicking the file name, your web browser opens instead of the HTML editor. To edit the HTML, either associate the htm file type with your HTML editor or open the file from within the editor.

Step 5: Customize the HTML Template (Optional)

- To view and edit the fields displayed on your web page:
- 1 Click **Customize HTML**.
The **Customize HTML** window is displayed, showing the ABO properties (fields) that will be generated for your HTML template. For information about this window, see **Customizing HTML Before Generation**, page 103.
 - 2 Customize the template as desired.
 - 3 Click **Close** to return to the **Configuration** window.
 - 4 Click **Next**.
The **Ready to generate** window is displayed.

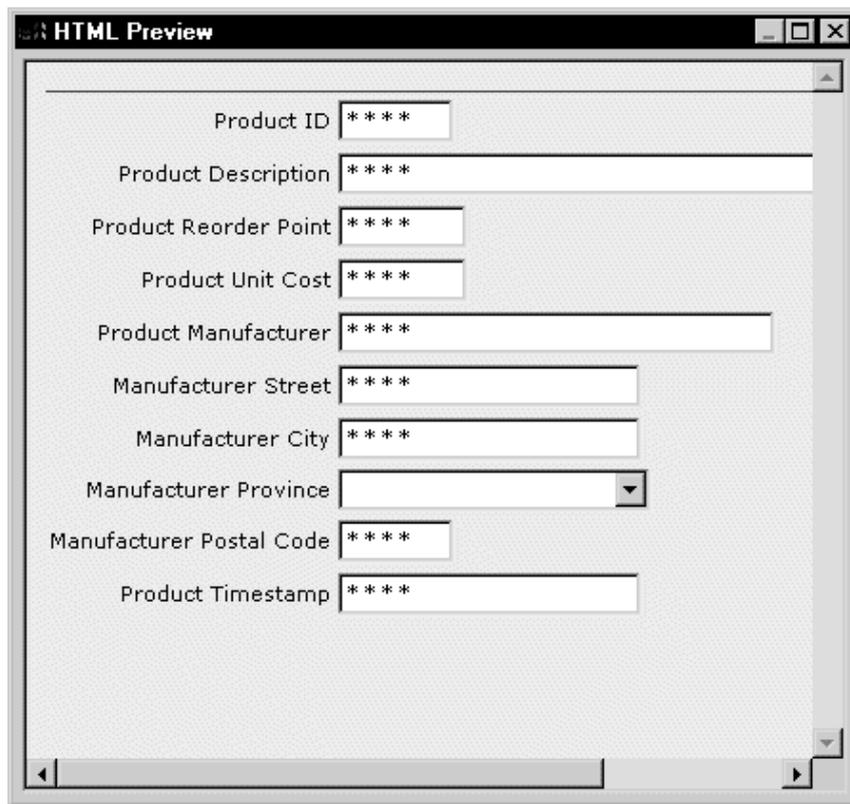
Step 6: Generate the HTML Template



HTML Template Wizard — Ready to Generate Window

Using this window, you can perform the following actions:

- To view the generate report, click Generate.
If you have a code comparison utility installed and configured for use with Construct Spectrum, you can also compare the new code with code from an earlier generation.
 - For information about using a code comparison utility, see **Use Reports with a Code Comparison Tool**, page 67, *Construct Spectrum SDK Reference*.
 - For information about the generated report, see **Generating and Reviewing Reports**, page 63, *Construct Spectrum SDK Reference*.
- To view your HTML code as it will appear in a web browser, click Preview:



The screenshot shows a window titled "HTML Preview" with a form containing the following fields:

Product ID	****
Product Description	****
Product Reorder Point	****
Product Unit Cost	****
Product Manufacturer	****
Manufacturer Street	****
Manufacturer City	****
Manufacturer Province	▼
Manufacturer Postal Code	****
Product Timestamp	****

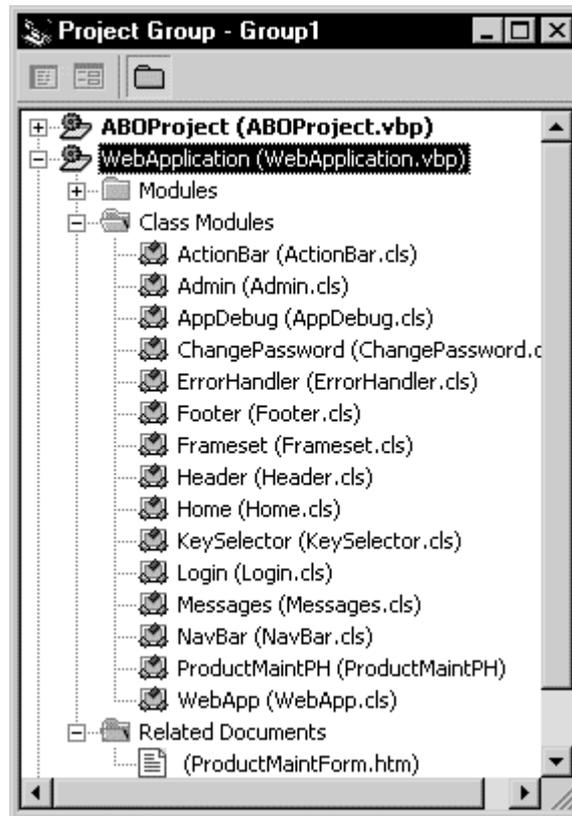
Preview of HTML Code

- To invoke the wizard again to generate another HTML template, select Yes.
- To generate the code, update the Visual Basic project, and close the wizard, click Finish.

When generation is complete, a message is displayed describing the success or failure of the operation. If there were problems with generation, the window prompts you to view the generated report.

After Generation is Complete

After generating the HTML template, it is saved to the directory where your web project is stored. If you selected Add as related document in the wizard, the HTML file is listed in the Related documents folder in Project Explorer (shown near the bottom of the Project Explorer in the following example):



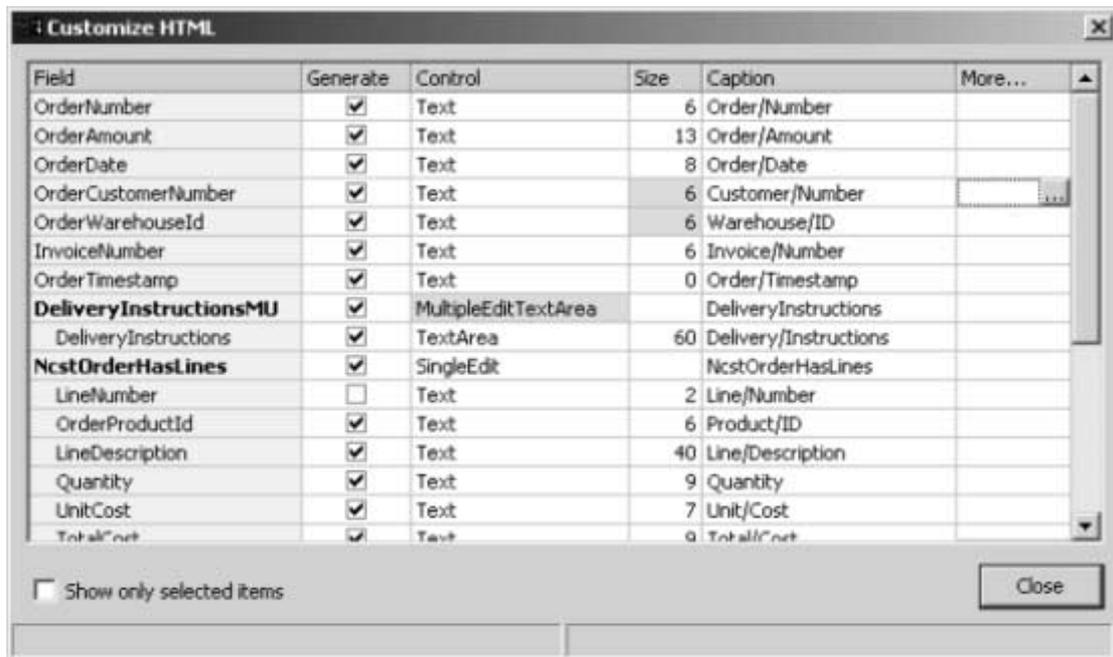
Project Explorer — Web Project after Generating an HTML Template

After creating an HTML template, you must update the project's object factory. You can then run the application, test, debug, and customize it. For information, see **Updating and Customizing the Object Factory**, page 137.

Customizing HTML Before Generation

- To customize the HTML template before generation:
 - 1 Display the Configuration editor for the HTML Template wizard.
 - 2 Click Customize HTML.
The Customize HTML window is displayed, showing the database fields included on your HTML page, as well as their characteristics. The window differs, depending on whether you are customizing a template for a maintenance or browse page.

Example of an HTML template for a maintenance page



Customize HTML Window for a Maintenance Page

Example of an HTML template for a browse page

Customize HTML Window for a Browse Page

Tip: Construct Spectrum keeps track of some changes you make to the fields. After you change a default value, the cell is highlighted. If you select the cell, the default value is displayed in the message area at the bottom of the window.

The following sections describe how to customize HTML templates.

Customizing a Maintenance Page

Using the Customize HTML window, you can perform the following customizations to the HTML template for a maintenance page:

- Deselect fields for generation
- Change the type of control for a field
- Change values in selection lists
- Change values in radio button groups
- Change the view options for sections
- Change the width of a text box or text area
- Change the control's caption
- Add links from fields to browse pages or other maintenance pages

The following sections describe how to perform these customizations, as well as how controls are derived for web applications.

Deselect Fields for Generation

In the Generate column, the check boxes are selected by default for all fields, unless they have the GUI_NULL keyword attached to them in Predict.

- To omit a field from the generated template:
- 1 Deselect the Generate check box for the field.

Note: If you deselect fields for generation and want to view only fields that are selected, click Show only selected items.

Change the Type of Control for a Field

When the Natural subprogram for a business object is generated, Natural Construct looks at the database fields in Predict and derives controls to represent the fields on the client. The process of deriving controls is based on the following:

- Multiple-valued (MU) fields, periodic groups (PE), and related files are generated as HTML tables called “sections”. MU fields are generated with the multiple edit view. PEs and related files are generated with the single edit view.
- If a GUI keyword is attached to the field, the keyword determines the type of control.
- If a table verification rule is attached to the field, the field is generated as a selection list.
- If the field is not associated with a GUI keyword or table verification rule, the field is generated as a text control.

The following table summarizes the controls that are derived:

Predict Equivalent	HTML Equivalent	Tag example	VB GUI Equivalent
GUI_TEXTBOX keyword or no keyword or verification rule	Text	<INPUT Type="Text" Value="123">	Textbox
GUI_OPTIONBUTTON keyword	Radio	<INPUT Type="Radio" Value="123">	OptionButton
GUI_COMBOBOX keyword or table verification rule	Selection list	<SELECT> <OPTION Value="123"> </SELECT>	ComboBox
GUI_CHECKBOX keyword	Checkbox	<INPUT Type="CheckBox" Value="X">	CheckBox
GUI_ALPHAMULTILINE keyword	TextArea	<TEXTAREA>123 </TEXTAREA>	Textbox (Multiline)
GUI_PROTECTED keyword	ReadOnly	No tag used	Label
PE, MU, or related file	Section	Table tags used	Grid

➤ To change the default control for a field:

- 1 Click its Control.
A drop-down list is displayed.
- 2 Select an option: Text, Checkbox, Select, or Radio, Read only, or Text area.

If you change the control to Select or Radio, you can specify options to populate the list. These options are described in the next sections.

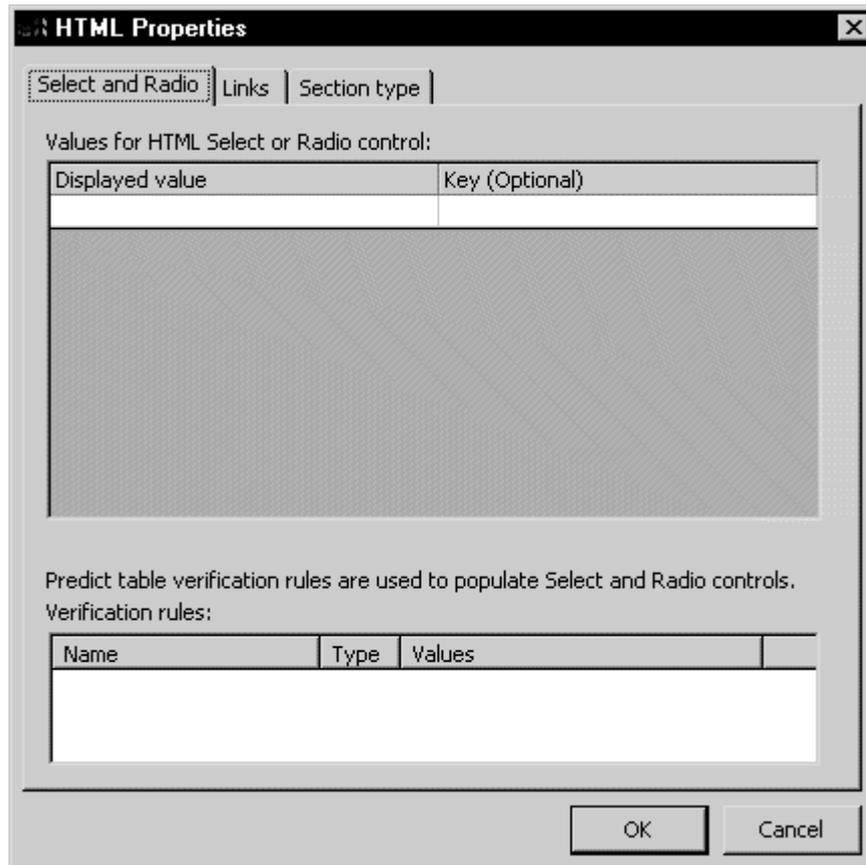
Specify or Modify Options on a Selection List

There are three ways a field can be represented by a selection list on the template:

- If the field has a table verification rule attached to it in Predict. In this case, you can view the rule and modify the options on the list.
- If the field has the GUI_COMBOBOX keyword attached to it in Predict. In this case, you can specify the options displayed on the list.
- If you changed the field's control to Select. In this case, you can specify the options displayed on the list.

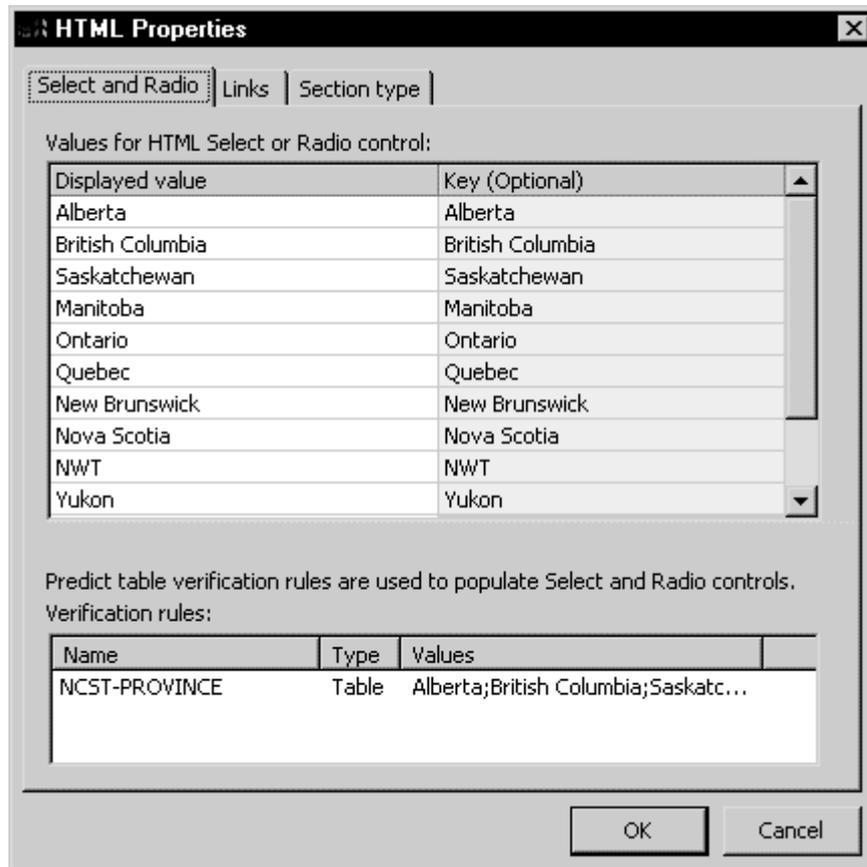
➤ To specify or modify options in a selection list:

- 1 Click the field's More cell.
The browse button is enabled.
- 2 Click the browse button.
The HTML Properties window is displayed, showing the Select and Radio tab:



HTML Properties — Select and Radio Tab

If the field has a table verification rule attached to it, the rule and values are displayed:



HTML Properties — Element Properties Showing Verification Rule

Tip: You can view verification rules attached to any field on the template, not just selection lists. This window shows all verification rules attached to a field.

- 3 Do one of the following:
 - If a table verification rule is shown, you can edit the options in the Displayed value column. However, you cannot remove them or specify additional values.
 - If you changed the field's control to a selection list, or the selection list is a result of the GUI_COMBOBOX keyword, specify the options you want to display in the Displayed value column. Press the Tab key to create new rows in the table.
 - If the displayed values must be changed before they can be returned to the database, specify the values in the Key (Optional) column.
- 4 Click OK to save your changes and close the HTML Properties window.

Specify or Modify Options in a Radio Button Group

There are three ways a field can be represented by a radio button group on the template:

- If the field has a table verification rule attached to it in Predict. In this case, you can view the rule and modify the radio buttons displayed in the group.
- If the field has the GUI_OPTIONBUTTON keyword attached to it in Predict. In this case, you can specify the radio buttons displayed in the group.
- If you changed the field's control to Radio. In this case, you can specify the radio buttons displayed in the group.

➤ To specify or modify radio buttons in a group:

- 1 Click the field's More cell.
The browse button is enabled.
- 2 Click the browse button.
The HTML Properties window is displayed, showing the Select and Radio tab.
If the field has a table verification rule attached to it, the rule and values are displayed.
For an example, see **Specify or Modify Options on a Selection List**, page 107.
- 3 Do one of the following:
 - If a table verification rule is shown, you can edit the displayed values. However, you cannot remove them or specify additional values.
 - If you have changed the field's control to Radio, or the radio button group is a result of the GUI_OPTIONBUTTON keyword, specify the options you want to display in the Displayed value column.
 - If the displayed options must be changed before they can be returned to the database, specify those values in the Key (Optional) column.
- 4 Click OK to save your changes and close the HTML Properties window.

Change View Options for Sections

Period groups (PE), multiple-valued fields (MU), and related files are generated as sections in an HTML template. You can present the content of sections in three ways: single edit, single edit with a View button that toggles to report view, and multiple edit view. In the case of an MU field with an alpha data type, you have an additional option: multiple edit text area.

Single Edit View

By default, sections are presented in single edit view:



Single Edit View

The user can view one record at a time and edit all fields on the record.

Single Edit View with Report View Option

In this case, a View button is displayed next to the section's title. The user can toggle between single edit view and report view, which shows multiple records:

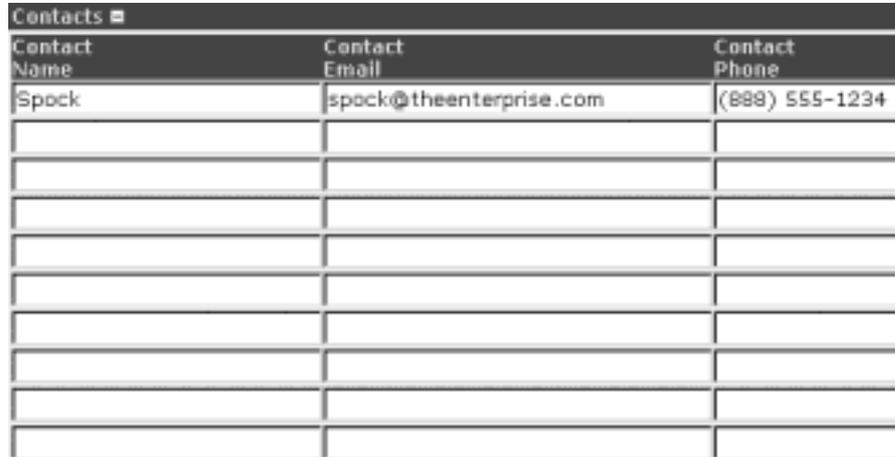
#	Cost Center	Acct	Project	Dist Amount
1	15	9342	AV	\$5,000.00
2	10	1068	AZ	\$5,000.00

Report View

In report view, the user cannot edit fields.

Multiple Edit View

Multiple edit view shows all records in the file in rows:



Contact Name	Contact Email	Contact Phone
Spock	spock@theenterprise.com	(888) 555-1234

Multiple Edit View

The user can edit any field.

Multiple Edit Text Area View

In this view, the section is presented as a text area in which the user can add or edit text:



DeliveryInstructions

Multiple Edit Text Area View

You can change the default size of the text area by modifying the cell dimensions in your generated HTML.

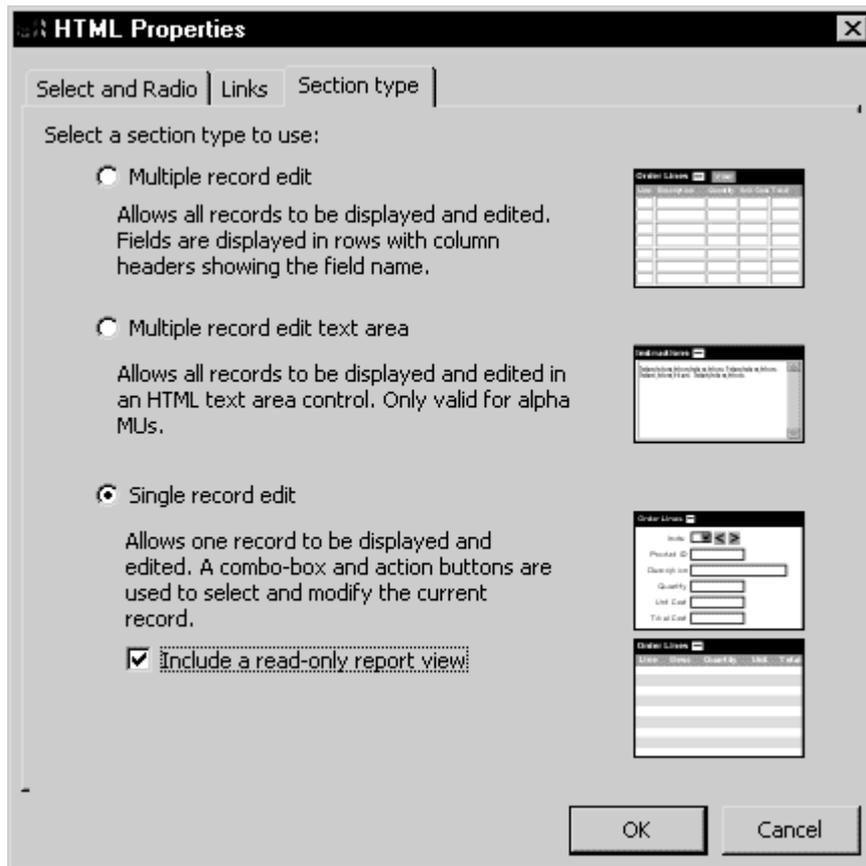
Collapsible Sections

Web page sections are collapsible. Users can close and reopen sections by clicking the minus (-) and plus (+) icons, respectively. This functionality saves space on the page and reduces the need for scrolling.

For information about supported versions of Microsoft Internet Explorer and Netscape Navigator, see **Supported Web Browsers**, page 22.

Change View Options

- To change a section's view options, do one of the following:
- 1 Click the section's Control cell.
A drop-down list is displayed.
 - 2 Select an option: Single Edit, Single Edit and Report, Multiple Edit, or Multiple Edit Text Area.
- or
- 1 Click the section's More cell.
The browse button is enabled.
 - 2 Click the browse button.
The HTML Properties window is displayed.
 - 3 Select the Section type tab:



HTML Properties — Section Type Tab

- 4 Select an option.
- 5 Click OK to save your change and close the window.

Change the Width of a Text Box or Text Area

The default width for a control is displayed in the Size column. However, you can only change the width of a text box or text area.

- To change the width of the control:
- 1 Replace the value in the field's Size cell.

Note: The size of the field as defined in Predict limits the amount of data that can be stored in the database field. If the user types in more characters than can be accepted, the value is truncated when the record is updated.

Change the Control Caption

The default caption for each field is determined by the field name stored in Predict or the name as modified in the ABO. You can change the caption for the field's control.

- To change the caption text:
- 1 Replace the value in the Caption or Header column.

Add a Link to a Browse Page

If your application includes an ABO and page handler for a browse object, you can create a link from a field on a maintenance page to the browse page. Users can click the Find button next to the field to open the drill-down page, where they can select a value and then return to the maintenance page. The selected value is displayed in the original field on the maintenance page.

If you are creating a maintenance page for an Order object, for example, one of the fields on the page may be Warehouse ID, which is also a field in the Warehouse browse object. The following illustration shows an excerpt from the Order maintenance page with a Find button next to Warehouse ID:

The screenshot shows a web form titled "Order Maint". It contains several input fields with "Find" buttons next to them:

- Order Number: Find
- Order Amount:
- Order Date:
- Customer Number: Find
- Warehouse ID: Find
- Invoice Number:

Order Maint Page

To display the Warehouse browse page, the user clicks the Find button:

The screenshot shows a web form titled "Warehouse Browse". It contains the following search fields:

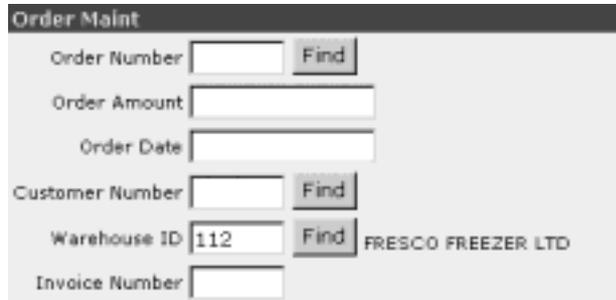
- Sort Key: WarehouseId (dropdown)
- WarehouseIdKey:
- Range: + (dropdown)
- Go:

Below the search fields is a table with the following data:

#	ID	Description	Street	City	Province	Postal
1	111	TORONTO CENTRAL WAREHOUSE	1120 JARVIS ST	TORONTO	Ontario	N1M 2E5
2	112	FRESCO FREEZER LTD	7869 OTTAWA ST.	TORONTO	Ontario	M4P 1E1
3	113	SOUTHERN DISTRIBUTORS LIMITED	6430 CHARLES ST	WINDSOR	Manitoba	J1M 2E5
4	134	MONTREAL CENTRAL WAREHOUSE	1120 JARVIS ST	MONTREAL	Ontario	M1M 2E5

Warehouse Browse Page

To select a warehouse, the user clicks the red arrow next to a warehouse ID. The maintenance page is redisplayed, showing the selected warehouse ID:



The screenshot shows a form titled "Order Maint" with several input fields and buttons. The fields are: Order Number, Order Amount, Order Date, Customer Number, Warehouse ID, and Invoice Number. Each field has a "Find" button next to it. The "Warehouse ID" field contains the value "112" and the text "FRESCO FREEZER LTD" is displayed to its right.

Order Maint Page after Look Up

- To add a link from a maintenance field to a browse page:
 - 1 Click the field's More cell.
The browse button is enabled.
 - 2 Click the browse button.
The HTML Properties window is displayed.

- 3 Select the Links tab:

HTML Properties

Select and Radio **Links** Section type

Source Field: OrderCustomerNumber

You can link this field to another object in your application. Additional HTML will be generated, such as a Find button to invoke a browse page or an HTML link to a maintenance page.

Target Page Handler:
Customer2.Browse

Target Sort Key:

Display as Popup window.

Key Match:

Source	Target

Return Value(s):

Source	Target
CustomerNumber	OrderCustomerNumber
BusinessName	OrderBusinessName

OK Cancel

HTML Properties — Links Tab

- 4 Select the page handler ID of the browse page from Target Page Handler.

Note: If you want the browse page displayed as a pop-up window, ensure that Display as Popup window is selected.

- 5 Provide the return values for the link in Return Value(s).
In the example, CustomerNumber maps to OrderCustomerNumber and BusinessName maps to OrderBusinessName.
- 6 Click OK to save your changes and close the HTML Properties window.

Add a Link to a Maintenance Page

If your application includes an ABO and page handler for another maintenance object that corresponds to a maintenance page, you can create a link from a field on one maintenance page to another maintenance page. The user can click the link for a particular record on the maintenance page to open the corresponding maintenance page, where the selected record is displayed.

- To add a link from a field on a maintenance page to another maintenance page:
- 1 Click the field's More cell.
The browse button is enabled.
 - 2 Click the browse button.
The HTML Properties window is displayed.
 - 3 Select the Links tab.
The Links tab is displayed.
 - 4 Select the page handler ID of the target maintenance page from Target Page Handler.
 - 5 Provide the name of the field containing the primary key for the source and target pages in Key Match.
 - 6 Click OK to save your changes and close the HTML Properties window.
The HTML Template wizard supplies the return link from the maintenance page.

Customizing a Browse Page

Using the Customize HTML window, you can perform the following customizations to the HTML for a browse page:

- Deselect fields for generation
- Change the alignment of columns on the browse page
- Add links from browse fields to maintenance or other browse pages
- Change the headers for browse columns

The following sections describe how to perform these customizations.

Deselect Fields for Generation

In the Generate column, the check boxes are selected by default for all fields, unless they have the GUI_NULL keyword attached to them in Predict.

- To omit a field from the generated template:
- 1 Click the Generate check box to deselect the field.

If you have deselected fields and want to view only fields that are selected for generation, click Show only selected items.

Change the Alignment of a Column in a Browse Table

- To change the alignment of a column:
 - 1 Click the field's Alignment cell.
A drop-down list is displayed.
 - 2 Select an option: Left, Center, or Right.

Add a Link to a Maintenance Page

If your application includes an ABO and page handler for a maintenance object that corresponds to the browse object, you can create a link from a field on the browse page to a maintenance page. The user can click the link on a particular record on the browse page to open the maintenance page, where the selected record is displayed. The user can then edit the record on the maintenance page.

If you are creating a browse page for a Customer object, for example, you can link records on the browse page to records on the Customer maintenance page. The following illustration shows an excerpt from the Customer browse page in the demo web application. The Customer Number field is linked:



Customer Browse Page

- To add a link from a field on a browse page to a maintenance page:
 - 1 Click the field's More cell.
The browse button is enabled.
 - 2 Click the browse button.
The HTML Properties window is displayed.

- 3 Select the Links tab:

The screenshot shows the 'HTML Properties' dialog box with the 'Links' tab selected. The 'Source Field' is set to 'CustomerNumber'. Below this, there is explanatory text: 'You can link this field to another object in your application. Additional HTML will be generated, such as a Find button to invoke a browse page or an HTML link to a maintenance page.' The 'Target Page Handler' field contains 'Customer2.Maint'. The 'Target Sort Key' field is empty. There is an unchecked checkbox for 'Display as Popup window'. The 'Key Match' section contains a table with two columns: 'Source' and 'Target'. The first row shows 'CustomerNumber' in both columns. Below this is a 'Return Value(s)' section with an empty table with 'Source' and 'Target' columns. At the bottom are 'OK' and 'Cancel' buttons.

Source	Target
CustomerNumber	CustomerNumber

Source	Target

HTML Properties — Links Tab

- 4 Select the page handler ID for the maintenance page from Target Page Handler.
 - 5 Provide the name of the field containing the primary key for the source and target pages in Key Match.
 - 6 Click OK to save your changes and close the HTML Properties window.
- The HTML Template wizard supplies the return link from the maintenance page.

Add a Link to a Browse Page

If your application includes an ABO and page handler for another browse object that displays additional (drill-down) details for a browse object, you can create a link from the field on this browse page to the other browse page. The user can click the link on a particular record on the browse page to open the drill-down browse page, where the selected record is displayed.

- To add a link from a field on a browse page to another browse page:
 - 1 Click the field's More cell.
The browse button is enabled.
 - 2 Click the browse button.
The HTML Properties window is displayed.
 - 3 Select the Links tab.
The Links tab is displayed.
 - 4 Select the page handler ID of the browse page from Target Page Handler.
 - 5 Select the sort key for the target browse page from Target Sort Key.
The sort key is the name of the field used to sort records on the target browse page.
 - 6 Provide the name of the field containing the primary key for the source and target pages in Key Match.
 - 7 Click OK to save your changes and close the HTML Properties window.

Change Header Text

The default header for each column in the browse page is determined by the field name stored in Predict or the name as modified in the ABO. You can change the header text.

- To change the text:
 - 1 Replace the value in the Caption or Header column.

USING HTML REPLACEMENT TAGS

Special replacement indicator tags embedded in the HTML templates allow you to have “live” content presented programmatically on your web pages. This chapter describes the syntax of these tags, the tags supplied with Construct Spectrum, and how to create your own tags.

The following topics are covered:

- **How Page Handlers Process Replacement Tags**, page 122
- **Syntax for Replacement Tags**, page 123
- **Types of Replacement Tags**, page 123
- **Supplied Replacement Tags**, page 125
- **Defining Custom Replacement Tags**, page 135

How Page Handlers Process Replacement Tags

Page handlers interact with the ABO and HTML templates to gather information they need to assemble appropriate content for a web page. With the information available in the ABO, page handlers themselves require little detail to process information about any one field; page handler code is comparatively generic. The parse area for the page handler includes two separate specifications for replacement tag processing: one for custom tags processing and the other for standard tag processing.

Each time an HTML template is requested, the associated page handler:

- Reads the template and scans for replacement tags.
- For each tag found, the page handler:
 - calls the tag processing framework component for custom tags
 - performs specific processing within the template loop
 - performs replacements global to the application
- Replaces the tag with appropriate content.
- Sends content to the browser after all the replacements are made.

The prefix signals the page handler that it must find appropriate content to display on the requested web page. For example, consider the following replacement tag:

```
<!--cst:PAGE Page="Navbar" /-->
```

In this example, the page handler creates a page using a page handler called Navbar. When the page handler finds the replacement value, it stores it in a buffer and continues reading the template. If it finds another replacement tag, it stores this value as well. This process repeats until all replacement tags have been read. When all replacements have been stored in the buffer, the page handler sends the assembled content to the browser.

Syntax for Replacement Tags

The syntax for replacement tags combines HTML comment format with XML tags. The syntax allows attributes, requires end tags, and is supported by HTML editors.

Construct Spectrum replacement tags have the following syntax:

Comment indicator	Prefix	Tag Name	Attribute	End of tag
<!--	cst:	NAME	Attribute="Value"	--> or /-->

Tag names are case-sensitive; for example, FIELD is a different tag name than Field. Attribute information is not case-sensitive.

Note: All replacement tags supplied with Construct Spectrum use upper case for tag names.

Types of Replacement Tags

While all Construct Spectrum replacement tags follow the syntax described in the previous section, they differ in complexity to address typical replacement needs. Replacement tags can be categorized according to the following types:

- Simple
- Conditional
- Repeating
- Complex

Simple

Simple replacement tags retrieve a single value to display in the browser. These replacement tags are self-contained between the start and end tags.

Conditional

Conditional replacement tags require checking to determine whether to display the requested value. For example, the SECURITY tag checks whether the user has permission to retrieve the information requested. If the user has permission, the requested content is displayed in the browser; if not, blank lines are returned.

Repeating

Repeating replacement tags use loop processing. Such tags describe the appearance of one row. They specify the field name to use, the attributes for the field, and the number of times the process is performed. Substitutions are made for each repetition. Consider the following example:

```
<!--cst: REPEAT Control= "NumberOfRows"-->
  <!--cst:FIELD Name="CustNo (%1)"/-->
<!--/cst:REPEAT -->
```

In this example, the (%1) sets up a process to perform replacements for the number of rows specified in the subprogram code. When the page handler scans the REPEAT tag, it performs the replacement for the first line and then gets each consecutive line until it finds the last line number specified by the code.

Complex

Complex replacement tags permit greater flexibility in specifying the kinds and amounts of material to display. These replacement tags can combine, nest, or embed tags within the single structure or retrieve multiple values for display.

Consider the following replacement tag example from CustomerMaintForm.htm:

```
<!--cst:FIELD Name="CustomerNumber" Value="#VALUE"-->
  <INPUT TYPE="TEXT" SIZE=10 MAXLENGTH=10 NAME="CustomerNumber"
  Value="#VALUE">
<!--/cst:FIELD-->
```

In this example, the page handler finds the value for Customer Number and places it in the Value attribute of the INPUT tag.

Tip: While all tags can be nested, ensure that you place your end tags carefully. Overlapping tags result in XML formation errors and cause the tag processor to fail.

Supplied Replacement Tags

The following sections describe the function and attributes of the HTML replacement tags supplied with Construct Spectrum.

ALTERNATE

This tag is usually contained inside a REPEAT tag. It varies a value based on a counter. For example, if Freq=2, a value is changed every two times.

Attributes

Replace=*Indicator*

Value=*Alternate*

Other=*OtherAlternate*

Number=*CurrentValue*

Freq=*Frequency*

Example of the ALTERNATE tag

```
<!--cst:ALTERNATE Replace="#ALTCOLOR"
      Value="Red"
      Other="White"
      Number="%1" Freq="2"-->
  <TR BGCOLOR="#ALTCOLOR">%1</TR>
<!--/cst:ALTERNATE-->
```

BROWSE

This tag creates a link to a browse page. A client-side JavaScript adds attributes at runtime.

Attributes

Page=*PageHandlerID*

SortKey=*SortKeyField*

URL=*ReplacementURL*

Example of the BROWSE tag

```
<!--cst:BROWSE Page="Customer.Browse" SortKey="CustomerNumber"
      URL="#URL"-->
  <A HREF="#URL">Browse customers</A>
<!--/cst:BROWSE-->
```

BROWSER

This tag removes content if the browser is not at a specified level. The Type attribute indicates the required browser.

Attributes

Type=(Nav,IE)

Example of the BROWSER tag

```
<!--cst:BROWSER Type="IE"-->
  <A HREF="www.microsoft.com/IE">
    This is an IE-only link
  </A>
<!--/cst:BROWSER>
```

CHECKBOX

This tag adds the CHECKED tag to a CHECKBOX input tag. If the user turns on the checkbox, the Value attribute of the input tag returns an “X”.

Attributes

Field=*Fieldname*

Value=*ReplaceValue*

Example of the CHECKBOX tag

```
<!--cst:CHECKBOX Field="CancelOrder" Value="#VALUE"-->
  <INPUT TYPE="CHECKBOX" Value="X" Name="CancelOrder" #VALUE>
</INPUT>
<!--/cst:CHECKBOX-->
```

DRILL

Use this tag to “drill down” to additional details in a browse page to browse page linking operation.

ERROR

This tag enables the content between the start and end tags if the field specified in the Field attribute contains a validation error.

Note: For Spectrum V4.3.1 project templates, this tag is used for Netscape Navigator only.

Attributes

Field=*FieldName*

Example of the ERROR tag

```
<!--cst:ERROR Field="CustomerNo"-->
  <BOLD>Error in customer number
</BOLD>
<!--/cst:ERROR-->
```

ERRORS

This tag includes errors detected in the page handler. The Fields attribute specifies where to include a delimited list of field names that have errors. The Messages attribute specifies where to include a delimited list of error messages for those fields.

Attributes

Messages=*ReplaceMsg*

Fields=*ReplaceFields*

Example of the ERRORS tag

```
<!--cst:ERRORS Messages="#MSGs" Fields="#FIELDS"-->
  <INPUT Type="HIDDEN" Name="ErrorMsgs" Value="#MSGs">
  <INPUT Type="HIDDEN" Name="ErrorFields" Value="#FIELDS">
<!--/cst:ERRORS-->
```

FIELD

This tag inserts the value of a field (ABO property). When the Value attribute is present, the HTML between the start and end tags is scanned and the value for the field is substituted.

Attributes

Name=*Fieldname*, [Value=*Indicator*]

Examples of the FIELD tag

```
<!--cst:FIELD Name="CustomerNumber" /-->

<!--cst:FIELD Name="OrdNum" Value="#REPLACE"-->
  Number is #REPLACE
<!--/cst:FIELD-->
```

INDEX

This tag replaces all occurrences of %1 between the start and end tags with the value of the index specified in the Field attribute. It is used on maintenance pages for single edit sections.

Attributes

Field=*Fieldname*

Example of the INDEX tag

```
<!--cst:INDEX Name="OrderHasLines"-->
  The current index is:%1
  <!--cst:FIELD Name="OrderLn(%1)"/-->
<!--/cst:INDEX-->
```

INFRAME

This tag displays the content between the start and end tags if the application is in Frames mode.

Example of the INFRAME tag

```
<!--cst:INFRAME-->
  <HTML><BODY>
<!--/cst:INFRAME>
```

INSTANCE

This tag returns the instance ID for the current maintenance page handler. The instance ID ensures that only the current maintenance page is updated.

Example of the INSTANCE tag

```
<!--cst:INSTANCE/-->
```

LOGGEDIN

This tag displays the content between the start and end tags if the user is currently logged in.

Example of the LOGGEDIN tag

```
<!--cst:LOGGEDIN-->
  You are logged in.
<!--/cst:LOGGEDIN-->
```

LOGGEDOUT

This tag displays the content between the start and end tags if the user is not logged in.

Example of the LOGGEDOUT tag

```
<!--cst:LOGGEDOUT-->
  Click here to log in.
<!--/cst:LOGGEDOUT-->
```

LOOKUP

This tag starts a foreign key browse from a maintenance page.

Attributes

Page=*PageHandlerID*

URL=*ReplacementURL*

Example of the LOOKUP tag

```
<!--cst:LOOKUP Page="Customer.Browse" URL="#URL"-->
  <A HREF="#URL">Lookup customers</A>
<!--/cst:LOOKUP-->
```

LOOKUP_VALUES

This tag retrieves the lookup values specified in the request.

MAINT

This tag creates a link to a maintenance page, usually from browse rows.

Attributes

Page=PageHandlerID

Key=Keyfield

LookupValue=Value

Example of a MAINT tag

```
<!--cst:MAINT
  Page="Warehouse.Maint"
  Key="WarehouseIDm"
  LookupValue="WarehouseID(%1)"
  URL="#URL"-->
  <A HREF="#URL">Edit Warehouse</A>
<!--/cst:LOOKUP-->
```

MENU

This tag is used to build a flyout (pull-down) menu.

Attributes

MENU.XML=XML file

MENU.XSL=XSL file to transform xml file

MODIFY

This tag is used in a maintenance page to maintenance page linking operation.

NOFRAME

This tag contains the code generated when you are operating in Non-frames mode.

ONLOAD_ADD

The contents of this tag are added to the Body_Onload event for the page.

PAGE

This tag returns the contents of a page handler, usually a section of an HTML page.

Attributes

Handler=*PageHandlerID*, [Content=*ContentID*]

Example of the PAGE tag

```
<!--cst:PAGE Handler="Customer.Maint"/-->
```

PARENT_FIELDS

This tag retrieves the ParentFields property from the URL query string.

PARENT_FORM

This tag retrieves the ParentFormName property from the URL query string.

POPUP

This tag retrieves the information for a browse pop-up window from the URL query string.

RADIO

This tag adds the CHECKED tag to the correct RADIO input tag.

Attributes

Field=*Fieldname*

Example of the RADIO tag

```
<!--cst:RADIO Field="CreditRating"-->  
  <INPUT Type="RADIO" Value="AA">  
  <INPUT Type="RADIO" Value="AAA">  
<!--/cst:RADIO-->
```

REPEAT

This tag repeats the HTML between the start and end tags.

- The Control attribute is a variable name containing the number of repetitions. By default, the REPEAT tag replaces all occurrences of %1 with the current repeat index.
- Use the Field attribute to repeat the content based on a field in an ABO.

Attributes

Control=*ControlVariable*

or

Field=*Fieldname*

Example of the REPEAT tag

```
<!--cst:REPEAT Control="NumberOfRows"=>
  Row number: %1
  <!--cst:FIELD Name="CustNo(%1)"/-->
<!--/cst:REPEAT-->
```

SECURITY

This tag enables content based on the security check for the value specified in the Tag attribute. There are three types: Page Handler, ABO, and Spectrum Object.

Attributes

Tag=*SecurityTag*

Examples of the SECURITY tag

```
<!--cst:SECURITY Tag="PH:Customer.Maint"-->
  <A HREF="?Page=Customer.Maint">
    Display customer maint
  </A>
<!--/cst:SECURITY>

<!--cst:SECURITY Tag="ABO:Cust.Maint.Update"-->
  You have update rights.
<!--/cst:SECURITY-->
```

SELECT

This tag adds the SELECTED attribute to the correct option tag based on the value of the Field attribute. In the following example, if the Province field contains ONT, the word SELECTED is inserted in the option tag containing ONT.

Attributes

Field=*Fieldname*

Example of the SELECT tag

```
<!--cst:SELECT Field="Province"-->  
  <SELECT Name="Province">  
    <OPTION Value="ONT">Ontario  
    <OPTION Value="QUE">Quebec  
  </SELECT>  
<!--/cst:SELECT>
```

SESSION

This tag displays values from the ASP session for the user, including the user ID, login time, and request time.

SUBMIT

This tag returns a URL for a specific page handler. The page handler processes the action contained in the Action attribute, if specified.

Attributes

URL=*ReplacementURL*

[Page=*PageHandlerID*]

[Action=*Action*]

Example of the SUBMIT tag

```
<!--cst:SUBMIT Page="Customer.Maint" URL="#URL"-->  
  <A HREF="#URL">Update Customers</A>  
<!--/cst:SUBMIT-->
```

TITLE

This tag returns the title of the current page handler.

Example of the TITLE tag

```
<!--cst:TITLE/-->
```

UPDATEKF

This tag updates a foreign key lookup field.

UPDATEPARENT

This tag updates a parent field from a browse pop-up window.

Defining Custom Replacement Tags

You can customize your application with custom replacement tags to override those supplied with Construct Spectrum. Custom replacement tags let you add functionality without affecting the supplied replacement tags.

- To create custom replacement tags:
 - 1 Add custom tags to the HTML template where you want content replaced dynamically.
 - 2 Do one of the following:
 - To use the tags throughout an application, add code to the TagProcessing.bas file.
 - To use the tags for one page only, modify the ParseTemplate function for the page handler.

The following sections describe these options.

Add Code to the TagProcessing.bas File

To use custom replacement tags throughout the web application, add them to the TagProcessing.bas file. TagProcessing.bas is one of the frameworks components added to your web project and is stored with other application files. It contains user-defined tag overrides and customizations. The syntax for customization is:

```
Public Function ProcessCustomTags(ByVal TemplateTag As TemplateTag) As Boolean

    ' Contains user defined TemplateTag overrides and customizations. If
    ' this function returns true the TemplateTag will not be processed
    ' or replaced from this point on.

    Select Case TemplateTag.Name
    Case "MYTAG"
        TemplateTag.Contents = "This is my TemplateTag."
        ProcessCustomTags = True
    End Select

End Function
```

Modify the ParseTemplate Function for the Page Handler

To use custom replacement tags for one page only, modify the ParseTemplate function for the page handler. In the following example, a custom tag exit is added to the ParseTemplate function of a maintenance page handler (see the section in bold on the following page):

```

Private Function ParseTemplate(FileName As String) As String

    Dim breplaced As Boolean
    Dim icnt As Integer
    Dim sname As String
    Dim sopt As String
    Dim sval As String
    Dim svals() As String

    Dim tp As TemplateParser
    Dim tag As TemplateTag

    Set tp = CreateTemplateParser(FileName)
    Do While tp.GetNextTag(tag)

        breplaced = TagProcessing.ProcessCustomTags(tag)

        ' ** Page handler tag replacement.
        If Not breplaced Then
            Select Case tag.Name
                Case "REPEAT"
                    Select Case tag.Attributes("Field")
                        Case Else
                            breplaced = False
                    End Select
                Case "INDEX"
                    Select Case tag.Attributes("Field")
                        Case Else
                            breplaced = False
                    End Select
                Case "SELECT"
                    Select Case tag.Attributes("Field")
                        Case Else
                            breplaced = False
                    End Select

                    '<cst:EXIT Name="ParseTemplate.CustomTags">
                    Case "TOTAL_COST"
                        tag.Contents = m_ABOInterface.GetField("Cost") +
                            m_ABOInterface.GetField("Tax")
                        breplaced = True
                    Case "PICTURE_FILENAME"
                        tag.Contents = "Pictures\" &
                            m_ABOInterface.GetField("ProductID") & ".gif"
                        breplaced = True
                    '</cst:EXIT>

            End Select
        End If
        ' Standard tag replacement.
        If Not breplaced Then
            TagProcessing.ProcessStandardTags tag, m_RequestData
        End If
        tp.ReplaceCurrentTag

    Loop
    ParseTemplate = tp.Buffer

End Function

```

UPDATING AND CUSTOMIZING THE OBJECT FACTORY

This chapter describes when, why, and how to update your web application's object factory using the Object Factory wizard. It also describes how to customize your web application in user exits.

The following topics are covered:

- **Introduction**, page 138
- **Using the Object Factory Wizard**, page 139
- **User Exits in the Object Factory**, page 142

Introduction

Each Construct Spectrum web application contains a file called the Object Factory (Ofactory.bas). The object factory encapsulates the business objects in a web application, making the application aware of the maintenance and browse actions enabled for the objects.

In addition to instantiating ABOs, the object factory also checks the current user's security profile to determine what business objects, actions, and methods the user can access. You can also use the object factory to customize security for your applications by adding custom code to the appropriate user exits. For information, see **Securing Your Application**, page 149.

Note: Whenever you add or modify an ABO or page handler, you must update the object factory for the application.

Using the Object Factory Wizard

Use the Object Factory wizard to quickly update the object factory after generating, modifying, or regenerating page handlers, and before compiling your web application.

Note: You can also update the object factory by right-clicking `Ofactory.bas` in Project Explorer and selecting Regenerate from the shortcut menu.

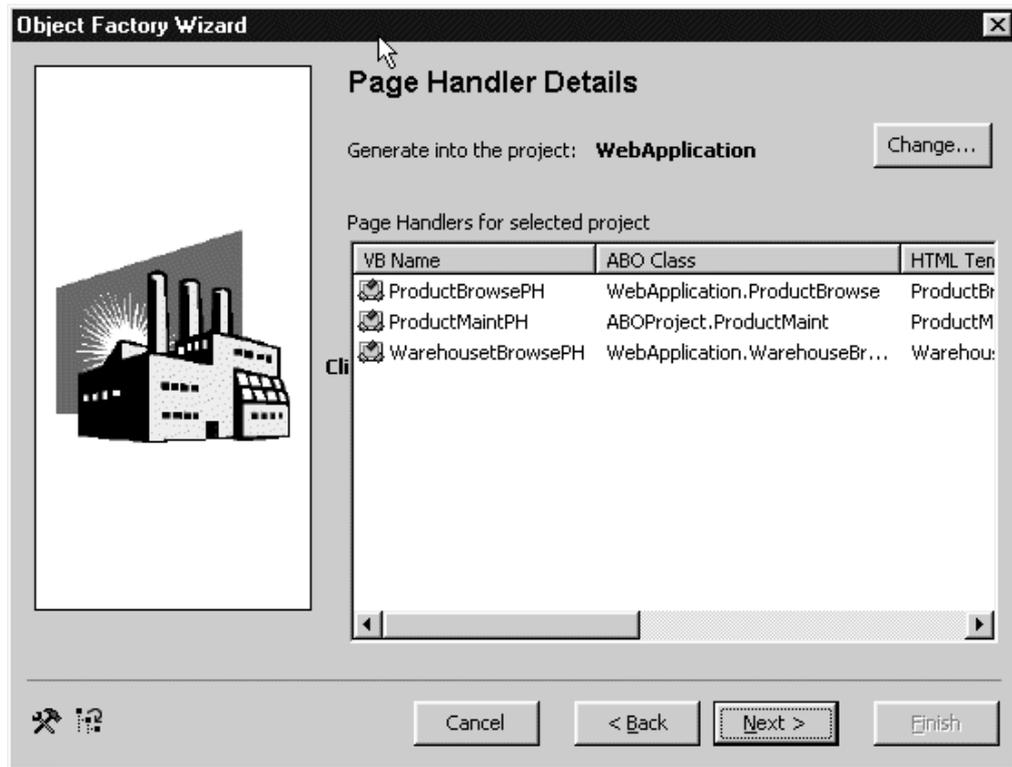
- To update your object factory:
- 1 Open the project group for your application in Visual Basic.
 - 2 Select Wizards > Object Factory from the Spectrum menu.
The Object Factory wizard is displayed:



Object Factory Wizard

For information about the Configuration editor or Spectrum Cache viewer (represented by icons in the lower left corner of the window), see **Features of the Wizards**, page 49, *Construct Spectrum SDK Reference*.

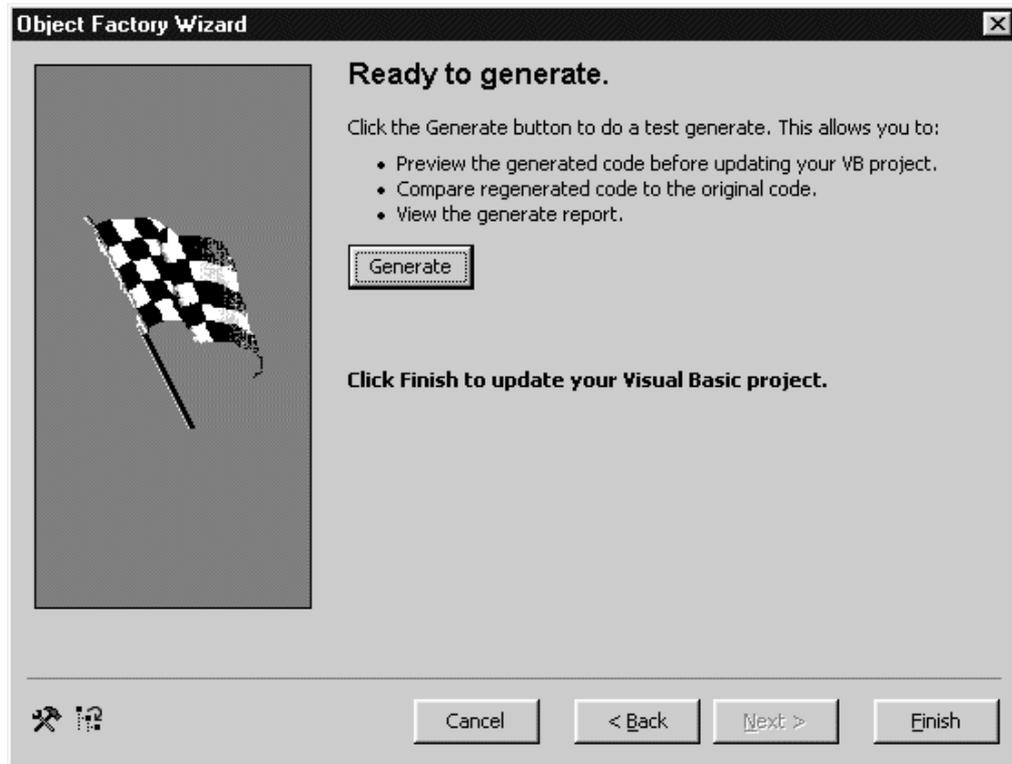
- 3 Click Next.
The Page Handler Details window is displayed:



Object Factory Wizard — Page Handler Details Window

- 4 Ensure that the correct web project is selected.
To change projects, click Change and select another project.

- 5 Click Next.
The Ready to Generate window is displayed:



Object Factory Wizard — Ready to Generate Window

- 6 At this point, you can perform two actions:
- To view the generated report, click Generate.
If you have a code comparison utility installed and configured for use with Construct Spectrum, you can also compare the new code you generated with code from an earlier generation of the module.

For information about using a code comparison utility with Construct Spectrum, see **Use Reports with a Code Comparison Tool**, page 67, *Construct Spectrum SDK Reference*. For information about the generate report, see **Generating and Reviewing Reports**, page 63, *Construct Spectrum SDK Reference*.
 - To generate the code, update the Visual Basic project, and close the wizard, click Finish.

When generation is complete, a message is displayed indicating the success or failure of the operation. If there were problems, you are prompted to view the generated report.

User Exits in the Object Factory

The object factory contains the `DefaultPage.SetDefault` user exit and three user exits you can use to define security options.

DefaultPage.SetDefault

This user exit allows you to change the default home page displayed when a user accesses a web application. This page is also displayed from the navigation bar using the Home link.

Security User Exits

You can add custom code to the security user exits in the object factory to customize or create your own security logic. For information, see **Securing Your Application**, page 149.

VALIDATING YOUR DATA

This chapter describes the data validation functionality provided with Construct Spectrum. It also describes how errors are displayed and handled in your web application.

The following topics are covered:

- **Types of Validations Used in Web Applications**, page 144
- **How Errors are Displayed on Web Pages**, page 145
- **Cached Errors on the Debug Page**, page 146

Types of Validations Used in Web Applications

Construct Spectrum implements four types of validation in web applications:

- BDTs based on logical formats in the ABO
- Validation routines in the ABO
- Validation routines in the page handler
- Predict verification rules defined for Natural subprograms

The following sections describe these types of validations in detail.

Business Data Types (BDTs)

BDTs present data to the user in a format that is consistent and based on business conventions rather than on programming language conventions. For example, a BDT can format a phone number with dashes (-) or a value that it is easily recognized by the user as associated with phone numbers. To do this, BDTs convert data values between simple internal Visual Basic data types and values that are displayed to the user in a browse or maintenance window. Construct Spectrum also uses BDTs to create sample strings to calculate the length of GUI controls.

Construct Spectrum includes a set of standard BDTs. You can use these BDTs as they are, customize them, or write your own.

For information about using and creating BDTs, see **Using Business Data Types (BDTs)**, page 121, *Construct Spectrum SDK Reference*.

For information about the `ICSTPageHandler_BDTOverrides` user exit to modify BDTs, see **Supplied User Exits**, page 84.

Validation Routines in the ABO

You can use user exits for the ABO wizard to code validations in the ABO. For example, you can provide validation code before setting a property value (Property Let function). For information about using implied user exits in the ABO, see **Working with Code**, page 58, *Construct Spectrum SDK Reference*.

The advantage of adding validation code to the ABO rather than to the page handler is that the validations are available to other page handlers. For more information, see **Using ActiveX Business Objects**, page 85, *Construct Spectrum SDK Reference*.

Validation Routines in the Page Handler

The Page Handler wizard provides a client validations user exit in which you can code validation routines that are specific to a web page. For information, see **PerformAction.ClientValidations**, page 87.

Predict Verification Rules

You can use the Predict verification rules defined for Natural subprograms to validate data. For example, you can generate values for a combo box based on Predict verification rules. Any verification errors in Natural subprograms are displayed on the web page.

For information, see **Verification Rules**, page 702, *Natural Construct Generation*.

How Errors are Displayed on Web Pages

When an error occurs in a Construct Spectrum web application, the affected text boxes are highlighted and an explanation of the error is displayed in the message area. To resolve the error, the user must clear the text box or provide the correct information.

The following example from the demo web application shows an excerpt from the Customer Maintenance page with two BDT errors:

Shipping Address

Street P.O. BOX 9

City Cambridge

Province Saskatchewan

Postal Code XXX

Contacts

Credit Rating AAA AA A B C D E

Credit Limit \$2,P00.00

Discount % 5

Warehouse ID 524 Find

Read Next Clear Add Update Delete

SPostalCode: Invalid postal code (XXX)
CreditLimit: Invalid currency value (\$2,P00.00)

Example of Errors in a Web Application

Cached Errors on the Debug Page

The Debug page provides global and session information that is useful for debugging applications. It also shows cached errors, which allows you to verify the application's response to validations. To open the Debug page, click Debug on the navigation bar of your web application.

The following example shows information gathered from the Customer maintenance page in the demo application:

Application Key	Value
Dispatcher.id	DISPATCH-431
Keep.sessiondata	True
Session Key	Value
~browser.level	1
~current.pageid	AppDebug
~messages	
~request.url	Page=Customer2.Maint
Customer2.maint.errors	Invalid postal code~XXX~SPostalCode;Invalid currency value~\$2,000.00~CreditLimit;
~userid	USER X
~password	
~session.time	9/27/99 4:35:26 PM
~refresh.frames	1
Customer2.maint.cache	1372 bytes
Customer2.maint.id	1
Customer2.maint.struct1900.control	1

Debug Page

This example shows the following information specific to the object:

Key	Value
CustomerMaint.cache	Displays the size of the cache in bytes.
CustomerMaint.id	Displays the unique ID of this instance of the page. For example, if you have two copies of the web browser open showing the same page, this field indicates the page's identifier in the cache.
CustomerMaint.errors	Lists the errors detected while using the object. You can use this field to check the errors that were generated as your page was being used.

Note: The error information on the Debug page is deleted when the error is resolved or when the application is closed.

For more information, see **Debug Page**, page 44.

SECURING YOUR APPLICATION

This chapter describes how to use the security facilities supported by Construct Spectrum and Microsoft Internet Information Server (IIS) to secure your web application. This chapter also describes how to use the user exits for the object factory and the Globals.bas module to modify security logic and functionality in Visual Basic.

The following topics are covered:

- **Using Construct Spectrum Security**, page 150
- **Using Microsoft Internet Information Server Security**, page 153

Using Construct Spectrum Security

Construct Spectrum supplies three levels of security:

- Construct Spectrum Administration subsystem
- SSL (secure sockets layer) encryption
- Secure login functionality

Spectrum Security

Using the Construct Spectrum Administration subsystem on the mainframe, you can define users, groups, and domains and then define user access privileges. This allows you to control user access to business objects and methods. You can also use Spectrum security in conjunction with Natural security or EntireX security.

For information about users and groups, see **Defining Groups and Users**, page 75, *Construct Spectrum Administration*.

Secure Sockets Layer (SSL)

If data sent between the web browser and web server must be encrypted, Construct Spectrum supports the Internet standard secure sockets layer (SSL) to encrypt messages.

Secure Login Functionality

The Login page is supplied with Construct Spectrum web application as a framework component. It uses HTTP security to authenticate the user ID and password.

Construct Spectrum web applications also support securing the Login page using a secure web protocol (HTTPS) while allowing the rest of the application to use a normal unsecured protocol (HTTP). This ensures that a user ID and password are encrypted when sent over the network when the user logs on. By default, new web applications are not set up to use this mode. However, you can enable the secure login functionality by modifying the `Globals.bas` module. For more information, see **Customizing the Globals.bas File**, page 152.

You can also establish login functionality as you generate page handlers using the Page Handler wizard. In the Configuration editor, select User must Login to restrict access to specific pages unless the user is logged onto the application. If a user is not logged on and requests a page handler that requires a login, the user is immediately redirected to the Login page.

Customizing Security

The following sections describe how to customize security logic in Visual Basic using the object factory and the Global.bas framework files.

Coding User Exits in the Object Factory

To customize login functionality and logic, modify and update the object factory. This involves adding custom code to an appropriate user exit, such as:

- IsPermitted.Override
- ValidateUser.Override
- IsPermitted.CustomSecurityTags

IsPermitted.Override

This user exit is part of the object factory's IsPermitted function and is used in conjunction with login settings. It resolves security tags on HTML templates. Use this user exit to override the default tag functionality. For example, as you refine your application, you may want to disable the IsPermitted function to bypass security and view the contents of all security tags. To do so, add the following code:

```
<CST:EXIT Name="IsPermitted.Override">
    IsPermitted=True
    EXIT FUNCTION
</cst:EXIT>
```

ValidateUser.Override

The Validate User function validates a user ID and password and creates a security profile. The security profile contains a list of Spectrum domains, objects, and methods the user can access. To have your application automatically validate a user, add the following code to this user exit:

```
ValidateUser=True
    "Exit Function
</cst:EXIT>
```

IsPermittedCustomTags

Use this user exit to create your own security tags. For example, you can add code similar to the following to create a security tag that only lets users with an Admin user ID view specific information:

```
<cst:EXIT Name="IsPermitted.CustomSecurityTags">
Case    "Custom.Admin"
        IsPermitted=(RequestData.UserID="Admin")
</cst:EXIT>
```

Customizing the Globals.bas File

To enable secure login functionality, add the following line of code to the Globals.bas framework module:

```
Public Const USE_SECURE_LOGIN = True
```

To enable an automatic login, you must also install a security key on your web server. You can obtain a key from a certificate authority and install it using the Key Manager in the Microsoft Management Console. For information about security keys, see **Key Manager**, page 153.

Using Microsoft Internet Information Server Security

The following sections provide an overview of the security facilities provided by the web server.

Key Manager

Use the Key Manager to create and manage the SSL key pair files that are necessary to establish encrypted communication lines with remote users. Also use the Key Manager to generate a request for a server certificate, which is a digital identification file that accesses the SSL key pairs. Without installing and attaching a valid server certification to your key pair, you cannot use your server's SSL security features.

Server Certificate

A server certificate contains identification information about you or the organization responsible for the content of a web site. The server certificate provides a way wherein users can authenticate your Web site and establish an SSL secured communication link. You can obtain a server certificate from a mutually trusted, third-party organization, called a certificate authority (for example, VeriSign). You must provide a detailed identification information before issuing a valid server certificate. After you receive a certificate file, use the Key Manager to install the server certificate.

SSL Key Pair

Use the Key Manager to create an SSL key pair, which is necessary for establishing a secure communication link. The key pair consists of a file called a public key and a private key. The key pair establishes a secure SSL connection with a user's web browser, not to encrypt transmitted information.

DEPLOYING YOUR APPLICATION

There are three methods you can use to deploy your Construct Spectrum web application: manually, using the Package and Deployment wizard in Visual Basic, or using third party software. This chapter describes how to deploy your application manually or using the Package and Deployment wizard in Visual Basic.

The following topics are covered:

- **Before Deploying the Application**, page 156
- **Deploying Manually**, page 156
- **Using the Package and Deployment Wizard**, page 159

Before Deploying the Application

Before deploying your application, ensure that you have:

- Compiled and saved both your ABO and web projects.
- Installed Construct Spectrum on the web server.

Deploying Manually

- To manually deploy your ABO and web applications:
 - ❑ **Step 1: Create ActiveX/COM Dynamic Link Libraries (DLLs)**, page 156
 - ❑ **Step 2: Register the Web DLL**, page 157
 - ❑ **Step 3: Create a Virtual Directory (Optional)**, page 157
 - ❑ **Step 4: Create a Starting Point for the Application**, page 157
 - ❑ **Step 5: Modify Application Settings**, page 158

Step 1: Create ActiveX/COM Dynamic Link Libraries (DLLs)

- To create ActiveX/COM dynamic link libraries (DLLs):
 - 1 Compile the ABO and web projects.
To compile the projects, select Make<Application name>.dll. from the File menu in Visual Basic.
 - 2 Collect the DLLs and support files and copy them to the web server.
 - 3 Use regsrv32.exp to register your ABO DLLs. For example:

```
regsrv32 ABO.dll
```

Step 2: Register the Web DLL

To register the web DLL, first create a new package for the Spectrum web DLL and then add the web application DLL to the new package and install the application.

- To create a new package for the Spectrum web DLL:
 - 1 Open the Microsoft Management Console.
 - 2 Right-click the Microsoft Transaction Server folder.
The shortcut menu is displayed.
 - 3 Select New > Folder from the shortcut menu.
 - 4 Provide a name for the new package.
This is the package in which you will insert the Spectrum DLL.

Step 3: Create a Virtual Directory (Optional)

Note: You only need to perform this step if you have not saved your projects under C:\inetpub\wwwroot.

- To create a virtual directory:
 - 1 Open the Microsoft Management Console.
 - 2 Right-click the Microsoft Internet Information Server folder.
The shortcut menu is displayed.
 - 3 Select New > Virtual Directory from the shortcut menu.
 - 4 Provide a name for the new folder and a link to your application folder.

Step 4: Create a Starting Point for the Application

- To create a starting point for the application:
 - 1 Open the Microsoft Management Console.
 - 2 Find your application directory in the Microsoft Internet Information Server folder.
 - 3 Right-click your project folder.
The shortcut menu is displayed.
 - 4 Select Properties from the shortcut menu.
The <Application Name> Properties window is displayed.
 - 5 Click Create to provide a starting point for the application.

Step 5: Modify Application Settings

This step applies to the web site manager.

- To modify the application settings:
 - 1 Open the Microsoft Management Console.
 - 2 Find your application directory in the Microsoft Internet Information Server folder.
 - 3 Right-click the application directory.
The shortcut menu is displayed.
 - 4 Select Properties from the shortcut menu.
Ensure you are on the Directory tab.
 - 5 Click Create.
 - 6 Select Run in separate memory space to run the application in a separate process from the web server process.
Running an isolated application protects other applications, including the web server, from being affected if this application fails.

Note: On Windows 2000 or XP, ensure that Application Protection is specified.

- 7 Click Configuration.
- 8 Select the App Options tab.
- 9 Select Enable session state to enable or disable session state.
When session state is enabled, Active Server Pages create a session for each user who accesses an ASP application so you can identify the user across pages in the application.

Note: If Session state is disabled, your application will not work.

- 10 Provide a time-out period.
If you specify a high value, the user will not have to log back on again as frequently. However, performance may suffer if the application is heavily used.
For more information about this window, click Help to open Microsoft Help or check the Microsoft MSDN Knowledge Base.

Using the Package and Deployment Wizard

You can use the Microsoft Package and Deployment wizard in Visual Basic to deploy your web application. Note that while the wizard scans your project, it does not recognize all application files, such as the HTML templates. When you deploy the application package, manually add the following application components:

- Graphics folder
- Support files folder
- ASP files
- HTML files

Before You Begin

Before using the wizard, ensure that you have:

- Compiled your ABO project to create a DLL.
- Customized application and ASP file settings. For information, see **Step 5: Modify Application Settings**, page 158.
- Prepared the web server by installing Construct Spectrum.
- Run Microsoft Transaction Server to register the web DLLs.

For more information about using the Package and Deployment wizard, refer to the online Microsoft knowledge base or the Windows help in Visual Basic.

Create the Distributable Package

Before you can deploy your web application, you must first bundle it into a distributable package comprised of one or more CAB files.

While creating the package, remember to:

- Specify Internet as the packaging script.
- Specify Internet as the package type.
- Specify C:\Inetpub\wwwroot as the Package Folder.
- Ensure the ABO and web DLLs are selected in the Included Files window.
- Ensure that the System, Spectrum runtime, and the Visual Basic runtime DLLs are not selected in the Included Files window.

Deploy the Package

After the application package is created, click **Deploy** on the **Package and Deployment** wizard to transfer your files to a web server. The wizard uses the HTTP Post method to transfer files to a Web server. For your web server to receive files from the **Package and Deployment** wizard, install the **Microsoft Posting Acceptor**. For information on installing the **Posting Acceptor**, read article Q192116 in the **Microsoft Knowledge Base**.

While deploying the package, remember to:

- Specify **Web publishing** as the deployment method.
- Select the application **CAB** and **HTML** files to deploy.
- Select the following components in the **Items to Deploy** window:
 - **Graphics** folder
 - **Support** folder
 - **HTML** files
 - **ASP** files
- Specify **HTTP Post** as the publishing protocol.

TIPS WHEN USING OTHER PROGRAMMING LANGUAGES

This chapter contains several tips and techniques you can consult when using other programming languages with the Construct Spectrum SDK to create web applications. This information is meant as a reminder when programming in a language other than Natural. For detailed information on each of these languages, see the respective user documentation.

The following topics are covered:

- **Using Javascript**, page 162
- **Using XSL**, page 163
- **Using HTML**, page 164

Using Javascript

This section contains helpful information when using Javascript.

Using the Equal Sign to Test for Equality

If `#a = 1` assigns the value of 1 to `#a`. To test for equality, use `If #a == 1`.

Indicating Comments in Code

- To comment out a portion of a line of code, use `/*` at the beginning of the portion and `*/` at the end.
- To comment out an entire line of code, use `//` at the beginning of the line.

Substituting Variables

If you want the value of a variable to be used at runtime, use the `eval` function. For example, you can use the following syntax to create dynamic names in a For loop:

```
NcstOrderLineName = 'document.all.ncstOrderLines' + i + '.value';
```

where `i` is the index in the For loop.

To use `document.all.ncstOrderLines1.value` as part of the XML syntax, use `eval(NcstOrderLineName)` in the javascript.

Using XSL

This section contains helpful information when using XSL.

Evaluating the Contents of a Variable

To evaluate the contents of a variable, add “\$” to the beginning of the variable name. For example:

```
<xsl:if test="$orderLines >= position()">
```

Incrementing a Variable in an XSL Loop

The XSL `position()` variable is automatically incremented in an XSL loop. You can use this variable for any purpose. For example, you can use it to test the position in the loop:

```
<xsl:if test="$orderLines >= position()">
```

Updating a Web Page Automatically

When you change XSL that is currently being referenced on a Web page, the Web page is automatically updated.

Using HTML

This section contains helpful information when using HTML.

Substituting Variables

To substitute the values for a variable, use { } to wrap the variable name. For example:

```
onfocus="displayDist({position()});
```

APPENDIX A: GLOSSARY

The following terms are used throughout the Construct Spectrum documentation set. Each term is listed with its meaning:

Term	Definition
active server page (ASP) script	Script that activates the WebApp.cls page handler, which opens the specified web page.
ActiveX business object (ABO)	Visual Basic class that represents a Natural business object on the client. The ABO wraps the Spectrum calls required to communicate with the Natural subprogram exposed by a subprogram proxy.
ActiveX DLL	Dynamic link library (DLL) containing one or more ABOs. It is used to package and deploy web applications.
application library	Natural library containing the server application components of a client/server application.
application service definition	Definition in the Construct Spectrum Administration subsystem that identifies the methods exposed by a subprogram. The definition is created automatically by the Subprogram-Proxy model. You can modify these settings on the Maintain Application Service Definition panel in the Construct Spectrum Administration subsystem.
application services	Natural subprogram implementing methods that can be called as remote services.
architecture	High-level description of the organization of functional responsibilities within a system. The architecture conveys information about the general structure of systems. It defines relationships between system components, but not the implementation of components.
browse command handler	Defines the commands linked to a browse dialog. It also acts as the initial target of commands, typically redirecting them to other application components. See also command handler , page 167.
browse data cache	Area containing database records returned from the server. Records are usually displayed in a browse dialog.

Term	Definition (continued)
browse dialog	Generic GUI browse dialog called to display any browse data residing on a mainframe or PC.
browse process	<p>Process by which framework components and generated browse components retrieve data and, optionally, display it in a browse dialog.</p> <p>For example, a browse process can retrieve rows of data, search for specific values, and then perform calculations and conditional processing. Users can display the results in a browse dialog, if desired.</p>
business data type (BDT)	<p>Type validation on the client that applies business semantics to a field. Typically, BDTs are used to format field data specified by the user.</p> <p>For example, if an application has an input field to enter a phone number, you can associate a BDT with the field to reformat the number with hyphens. A user can enter “7053332112”. When the user moves to the next field or performs another action, the number is automatically reformatted as 705-333-2112.</p> <p>Construct Spectrum supplies standard BDTs, which you can customize, or you can create your own. BDT modifiers are added to the keyword components of a field in Predict.</p>
BDT class	Collection of all BDT procedures.
BDT controller class	Collection of methods available to members of a BDT class. See also BDT class .
BDT controller object	Supplied client framework component that is an instance of the BDT controller class and uses the methods available to that class. Each application declares a BDT controller object, which records and maintains a list of names for each BDT and points to the BDT definition. See also business data type (BDT) , page 166.
BDT modifier	Additional logic users supply to modify the formatting or validation rules for a BDT. For example, <code>BTD_NUMERIC</code> ensures that only numeric values are entered in a field. You can also add a modifier to round numeric values. To increase flexibility, each BDT defines its own modifiers.
BDT procedure	Code that implements a BDT.

Term	Definition (continued)
business object	Conceptual abstraction that groups the attributes and behaviors associated with a business entity, such as Customer or Order. See also Visual Basic business object , page 179.
Business-Object-Super-Model	Model (available in the Construct Windows interface and Generation subsystem) that generates multiple modules for both web or client/server applications that do not use the Construct Spectrum client framework.
cardinality	Number of dimensions of information. Information with the same number of dimensions has the same cardinality.
child model	Individual model for which a super model (parent model) collects parameters and generates specifications.
client application	Portion of a Construct Spectrum client/server application that runs on a Windows platform.
client framework	Supplied set of cooperating Visual Basic classes that form a reusable design. It provides a skeleton of functionality, which you can customize or fill with generated and hand-coded Visual Basic modules. The client framework reduces the size of generated components and allows them to interact. It includes forms, classes, procedures, global variables, and constants that are shared among generated application components. It supplies both client and server components.
code block	One or more lines of code in a Visual Basic module that can be manipulated in the code editor as a block.
command block	Code block that tells the Natural Construct nucleus to treat the text within the block as a separate module and to apply the specified command to the block. Super models use command blocks to generate multiple modules.
command handler	Object, generally a Visual Basic class, that processes a command. The client framework calls command handlers when a user clicks a menu command or toolbar button. One command handler can handle multiple commands. See also command handler list , page 167 and hook , page 171.
command handler list	List of command handlers for each command ID. The last command handler hooked to a command ID is called first. See also hook , page 171.

Term	Definition (continued)
command ID	Unique identifier for an application-specific command sent when a user clicks a menu command or toolbar button. Define these commands by specifying a single command ID as “constant” for each unique menu and toolbar command.
complex redefine	Redefinition of a data area containing multiple data types, multiple redefinitions of a data field, or multiple levels of redefined fields.
compression	Reduce the byte size required to transmit data to and from the client and server. Data is compressed when it is sent and then decompressed when it reaches its destination. This reduces the size of data transmissions and improves network performance.
Construct Spectrum	Application consisting of a client and server component. The client component is a Construct Spectrum application running in Visual Basic. The server component is a set of subprograms accessed remotely by the client component.
Construct Spectrum Add-In	Customized functionality added to the Visual Basic environment.
Construct Spectrum Administration subsystem	Mainframe subsystem used to maintain and query tables defining Construct Spectrum application services and security.
database record	Logical view of database information. A database record can be comprised of one or more logically related database files or tables. Construct Spectrum represents database information in parameter data areas (PDAs).
DBID	Acronym for database ID, which is the number identifying the server database containing application components.
debugging tools	<p>Utilities you can use to locate and analyze logic errors. You can simulate client calls online and use traditional debugging tools, such as:</p> <ul style="list-style-type: none"> • Trace options, which allow you to save data from a client call to a file on the server and then use the data to recreate situations that caused errors. • Input and output statements, such as INPUT, PRINT, and WRITE, which allow you to step through the program for testing purposes. • Natural Debugging facility, which you can use to establish a debug environment. For information, see Natural Debugging Facility in the <i>Natural Utilities Manual</i>.

Term	Definition (continued)
dependent object	Object exposed by an OLE automation server that can only be created using the method of a higher-level object. See also externally-creatable object , page 169.
deployment	Movement of an application from a development environment to a production environment.
dialog	GUI form running on the client.
dispatcher or dispatch service	Server component used to broker communications between server components and client framework components. See also Spectrum dispatch service , page 177.
domain	Entity that defines a collection of related business objects (for example, Test, Admin, and Sales).
double-byte character set (DBCS)	Related collection of characters in some non-Latin languages that require two bytes to display.
download data	Transfer (copy) modules from the server to the client.
encapsulation	Technique in object-oriented programming in which the internal implementation details of an object are hidden from users of the object. Methods control how the object data is manipulated. Encapsulation allows internal implementations to change without affecting the way an object is used externally.
encryption	Encoding data so it is unusable for individuals without access to the decryption algorithms. Construct Spectrum allows you to encrypt sensitive data, such as payroll information, during network transmission. Data is decrypted when it reaches its destination.
Entire Broker service settings	Collection of Entire Broker-related parameters, including Entire Broker ID, server class, server name, and service.
Entire Broker stub	Entire Broker DLL on a Windows platform.
event	Action recognized by an object, such as pressing a key or clicking a mouse. You write code to respond to events.
externally-creatable object	Object exposed by an OLE automation server that can be created outside the server. See also dependent object , page 169.

Term	Definition (continued)
field	Component of a database record. The term also refers to areas on a panel in which values are entered.
flyout menu	Pull-down, cascading menu displaying other menus.
FNR	Acronym for the file number that identifies a specific server database file containing application components.
foreign key	Key field pointing to a record in an external file. For example, the demo application has an Order file containing a foreign key to the Warehouse field in the Warehouse file. Foreign keys can be set up with a browse function, enabling users to search for and select values.
form	<p>Dialog (window) that acts as the interface for an application. You add controls and graphics to a form to create the effect you want. Construct Spectrum supplies forms in the client framework and generates form modules for business object maintenance dialogs.</p> <p>When you run a project, forms are compiled into GUI dialogs that the user interacts with while using the application. Some forms, such as the generic BrowseDialog form, are dynamically configured at runtime by the client framework to alter the look of the form.</p> <p>Form definitions are saved in files with the extension .frm.</p>
form section	Portion of a web page containing a block of related information.
framework template	Structure or container supplied for applications. These customizable templates include header, footer, navigation bar, messages area, and constants.
generate	Process of producing code from specifications.
generated module	Generated component for either the client or server portion of an application. Generated server modules include Natural subprograms, subprogram proxies, and parameter data areas. Generated client modules include object factories, dialogs, and maintenance objects.
generation data cache	In-memory hierarchical data structure that allows you to quickly retrieve stored generation data.

Term	Definition (continued)
grid	<p>Displays 2-dimensional data for a client/server application in a table format.</p> <p>One-dimensional data shows one type of data, such as a phone number, name, or quantity. Two-dimensional data shows additional information in a grid or table. For example, the detail lines on a customer order can be displayed in a grid with each grid row corresponding to a unique line item. Each column in the grid corresponds to a discrete piece of information about the line, such as an item name, price, or quantity.</p>
grid control	<p>GUI control that displays related information in a table format. For example, purchase order line items can be displayed in a grid. The grid control supplied with Construct Spectrum sizes itself to the minimal width required to display all grid components. You can configure the supplied grid control as desired.</p>
group	<p>Collection of users defined in the Construct Spectrum Administration subsystem.</p>
GUI	<p>Acronym for graphical user interface.</p>
GUI control override	<p>Use Predict keywords to force a GUI control derivation. See also keyword, page 172.</p>
hook	<p>Associate a command handler object with a command ID. See also command handler, page 167, command handler list, page 167, and command ID, page 168.</p>
host	<p>See server, page 176.</p>
HTML fragment	<p>Portion of HTML that is not a complete web page.</p>
HTML template	<p>HTML that may contain replacement tags, which are dynamically exchanged for content or nested HTML templates at runtime.</p>
HTML Template wizard	<p>Wizard used to generate HTML templates.</p>
http request	<p>Parameterized list of named value pairs sent by a browser client to a web application.</p>
instantiation	<p>Process of creating an instance of a class. The result is an object.</p>

Term	Definition (continued)
internationalization	Adapting an application to make it easy to localize. See also localization , page 172.
job control language (JCL)	Command language used for batch jobs that tells the computer what to do.
keyword	Predict metadata type that acts as a label or identifier.
Level 1 data block	Level one field or structure and its subfields in a Natural parameter data area (PDA).
Level 1 data block optimization	Technique to improve the performance of client/server applications by reducing the volume of data transmitted across a network. Rather than sending all data blocks associated with an object, only the required blocks are sent.
library image file (LIF)	File that defines Natural definitions used by the Spectrum Dispatch Client.
LIF definitions module	BAS module in a Visual Basic project containing the definitions for application services, parameter data areas, and subprograms.
localization	Process of translating and adapting a software product for use in a different language or country.
lookups	Return descriptive information when a user requests a browse dialog or enters a value in a foreign key field on a maintenance dialog. For example, assume the Warehouse Number field is a foreign key field in the Order dialog and Warehouse Name is a descriptive field attached to the foreign key value. When a user enters a valid warehouse number, the lookup returns the name of the warehouse for display in the dialog.
maintenance dialog	GUI dialog from which a user can perform one or more actions on a business object. For example, a Customer Order object can be represented on a maintenance dialog. Using this dialog, an authorized user can add, delete, or update customer order information.
MDI child	Dialog (window) opened from an MDI parent dialog in a client/server application. For example, the Order maintenance dialog in the demo application is an MDI child to the MDI frame dialog.
MDI frame	Standard Visual Basic MDI frame supplied with the Construct Spectrum client framework.

Term	Definition (continued)
MDI parent	MDI dialog from which other dialogs are opened and displayed in a client/server application. The MDI frame supplied with the client framework is an MDI parent.
menu	<p>On a mainframe server, a panel or window listing available functions. To access a function, users enter a value in an input field or move the cursor to a value and press Enter.</p> <p>In Windows, a flyout (pull-down) listing of the available functions. To access a function, users select an option from the menu using the cursor or a keystroke combination.</p> <p>See also flyout menu, page 170.</p>
menu bar	Displays the menus available for user selection. By default, Construct Spectrum client/server applications contain File, Edit, Actions, Window, and Help menus on the menu bar, each containing standard menu commands.
metadata	Information about data. Metadata describes how physical data is formatted and interrelated. It includes descriptions of data elements, data files, and relationships between data entities. Typically, metadata is maintained in a repository known as a data dictionary, such as Predict.
method	Procedure that operates on an object and is implemented internally by the object. For example, the Update method updates a Customer Order object after changes to the order information.
model	Template used to generate modules. Each model contains one or more specification panels. Using these panels, you can specify parameters for a desired module and then generate the corresponding code. Natural Construct provides numerous models, including the Object-Maint-Subp and Subprogram-Proxy models.
module	Single application component, such as a hand-coded Natural program, subprogram, or data area or a Natural Construct-generated program, subprogram, data area, or subprogram proxy.
multi-level security	Security you can define at a high level or at a detailed level affecting many objects. For example, you can apply multi-level security to domains, objects, and methods.

Term	Definition (continued)
multiple-document interface (MDI)	Microsoft Windows paradigm for presenting windows whereby a parent window can encompass one or more child windows. See also MDI child , page 172, and MDI parent , page 173.
Natural Construct nucleus	Sophisticated driver program that invokes the model subprograms at the appropriate time in the generation process and performs functions common to all models, such as opening windows and performing PF-key functions. The nucleus communicates with the model subprograms through standard parameter data areas (PDAs). These PDAs contain fields assigned by Natural Construct, as well as fields required by a model.
Natural Debugging facility	Utility available in a Natural environment to help you locate and analyse logic errors. To access the facility, use the Invoke Proxy function in the Construct Spectrum Administration subsystem. The subprogram proxy sets up an online environment that simulates the client/server environment and allows you to use all the features of the Natural Debugging facility.
navigation bar	Menu bar on a web page containing links to other pages or actions.
node	Individual computer or, occasionally, another type of machine in a network.
nucleus	See Natural Construct nucleus , page 174.
object	Any application component, such as a form or record. A business object is a group of services related to a common business entity, such as Customer, Order, or Department.
object factory	Visual Basic module that identifies all objects and methods in an application and instantiates objects upon request.
Object Factory wizard	Visual Basic Add-In that updates an object factory in a Construct Spectrum web application.
object library	Provides definitions for all the objects, methods, and properties exposed by an OLE automation server. Equivalent to type library , page 178.
OLE	Acronym for object linking and embedding.
OLE automation server	Code component that passes objects to other applications so they can programmatically manipulate the objects.

Term	Definition (continued)
overflow condition	Situation where there are more fields than can be displayed on a dialog.
package	Collection of all modules necessary to implement a business object. A package combines components and classes to provide both browse and maintenance services for a database table. It is composed of a set of modules generated from a multi-module generation. An application is made up of one or more packages.
page handler	Visual Basic class that exchanges replacement tags on an HTML template with database content or another HTML template.
Page Handler wizard	Construct Spectrum Add-In that generates page handlers for web applications.
parent model	Super model that collects parameters for child models and generates specifications.
parse area	Code in a page handler that locates and exchanges HTML replacement tags.
ping	Request sent to a service to determine whether the service is running.
platform	Piece of equipment that, together with its operating system, serves as a base on which you can build other systems. For example, an MVS mainframe computer can serve as a platform for a large accounting system.
project	Collection of files used to build an application in Visual Basic.
project group	Collection of two or more Visual Basic projects, for example, web and ABO projects. A project group uses a .vbg extension.
Project window	Lists the forms and modules in the current Construct Spectrum project in Visual Basic.
Properties window	Lists the property settings for the selected form or control in Visual Basic.
property	Characteristic of an object, such as size, caption, or color. In Construct Spectrum, it refers to the data settings or attributes for an object in Visual Basic.

Term	Definition (continued)
regenerate/preserve status	Status indicating whether a code block in a module is regenerated or preserved during regeneration of the module. If you mark a block to be regenerated, it is replaced or deleted. If you mark a block to be preserved, it is not changed during regeneration.
remote call	Communication with an object residing in a different location, such as a server.
replacement tag	HTML tag that is replaced with database content or another HTML template when the web page is assembled. Some replacement tags can be used to remove existing sections of HTML. For example, you can use a security tag to specify content that only certain users can access.
resource	Text or binary value that can be localized. See also localization , page 172.
run	Execute or invoke a module or application.
security cache	File used to store recently-accessed security data.
server	Computer that provides services to another computer (called a client) and responds to requests for services. On multitasking machines, a process that provides services to another process is called a server.
server application	Application that runs on a server machine.
service	Software service that runs on a server. Several services can run on one server.
service exit	Exposed exit routine called by the Spectrum dispatch service; it can be replaced by a user-supplied routine.
service log	File used to store service log data.
shutdown	Command sent to a service to terminate the service.
software development kit (SDK)	See toolkit , page 178.
Spectrum client/server application	Application created using the Construct Spectrum wizards and add-ins. Users access mainframe business functions and data through a Visual Basic component running on a Windows platform.
Spectrum Control record	Record that is created daily and contains system control and statistic data for a Spectrum dispatch service.

Term	Definition (continued)
Spectrum Dispatch Client (SDC)	Provides the Construct Spectrum data exchange, which facilitates calls from a client to Natural subprograms running on a server.
Spectrum dispatch service	Middleware component that encapsulates broker calls on the server, provides directory services, enforces security, and invokes backend Natural services.
Spectrum security service	Component of the Construct Spectrum Administration subsystem that controls access to application libraries, objects, and methods.
Spectrum Service Manager	Client tool supplied with Construct Spectrum that allows you to specify which Spectrum services the client uses to communicate with the server.
Spectrum service settings	Collection of parameters used to configure a Spectrum service.
Spectrum web application	Application created using the Construct Spectrum wizards and add-ins. It allows users to access mainframe business functions and data from a web browser.
Spectrum web framework	Group of Visual Basic modules and classes that collaborate to dynamically generate web pages.
Status bar	Area that displays status information about a selected item, application, or business object in a client/server application. It contains sections for a message, status indicators, and the current date and time. Status bars are also displayed at the bottom of an MDI form.
steplib chain	Hierarchy of Natural libraries that determines the location from which modules are executed.
Sub Main procedure	First Visual Basic procedure executed when you run a Construct Spectrum application. Each Visual Basic application has one Sub Main procedure.
subprogram proxy	Natural subprogram called by a Spectrum dispatch service to translate data formats between the client and a Natural subprogram on the server. Each subprogram requires a subprogram proxy, which allows Construct Spectrum to provide a common interface to any subprogram.

Term	Definition (continued)
super model	Model that generates multiple components of a Construct Spectrum client/server or web application. Using a minimum number of input parameters, a super model determines the specifications for all models required to generate individual components of a package. See also package , page 175.
target module	See target subprogram , page 178.
target object	See target subprogram , page 178.
target subprogram	Any Natural subprogram.
template parser	Class used to parse HTML or other templates.
toolbar	Bar that provides quick access to commonly used commands in an application. A user clicks the appropriate toolbar button to perform the action it represents. Any action that can be performed from a toolbar can also be invoked from a menu.
toolbar button	Icon on a toolbar that allows users to perform an action.
toolkit	Set of related and reusable classes that provide general-purpose functionality. An application incorporates classes from one or more toolkits. Toolkits, or software development kits (SDKs), emphasize code reuse and are the object-oriented equivalent of subroutine libraries. For example, a toolkit can be a collection of classes for lists, associative tables, or stacks.
trace options	Options that specify how to trace messages sent between the client and server.
type library	Library containing definitions for all objects, methods, and properties exposed by the OLE automation server. See also object library , page 174.
upload data	Transfer modules from the client to the server.
variant	Visual Basic term identifying a late-binding data type. Variants allow Construct Spectrum subroutines or functions to accept different types of data. The exact type is determined when they receive the value in Visual Basic.
VB-Client-Server-Super-Model	Model that generates all modules required for a fully functional client/server application. The super model can generate all modules required for maintenance and browse services for up to 12 business objects at a time. See also super model , page 178.

Term	Definition (continued)
verification rule	<p>Predict-defined business rules that are implemented in the object subprogram on the server and the maintenance object on the client. They also provide default values for derived fields represented by GUI controls, such as check boxes, option buttons, or drop-down combo boxes.</p> <p>You can use verification rules to force users to make a selection based on one or more choices. For example, if an application has an input field for the state name, you can attach a verification rule to the field in Predict so that only valid state names are accepted.</p>
Visual Basic browse object	<p>Visual Basic class that configures an instance of a browse base class. This class delivers information about the columns and keys supported by the browse subprogram to the client framework, which configures and displays the browse dialog at runtime. See also Visual Basic business object, page 179.</p>
Visual Basic business object	<p>Conceptual browse or maintenance object comprised of class modules or objects with a domain on the client. It implements business rules and encapsulates communication with the Spectrum Dispatch Client (SDC).</p>
Visual Basic maintenance object	<p>Visual Basic class instantiated by a maintenance dialog to:</p> <ul style="list-style-type: none"> • encapsulate calls to the SDC • implement validation in the maintenance dialog <p>See also Visual Basic business object above.</p>
Web Super wizard	<p>Construct Spectrum Add-In to Visual Basic that generates multiple HTML templates and page handlers for a web application.</p>
web class	<p>Visual Basic class that responds to requests for a web page (ASP requests).</p>
web application ASP	<p>ASP (active server page) script used to instantiate a Spectrum web application.</p>

Term	Definition (continued)
wildcard	Character or symbol that qualifies a selection, such as “*”, “<”, or “>”. For example, using a value followed by an asterisk (*) indicates a range of file names beginning with that value. To list all modules that begin with “Maint”, enter “Maint*” as the selection criteria.
XML extract	Extract information from Predict and other sources, which is stored on the client as metadata in XML format. This includes information about business objects, as well as the formatting used by wizards to build application components. See also metadata , page 173.

INDEX

Symbols

- .css file
 - supplied
 - Styles.css, 67

A

- ABO
 - coding validations, 144
 - role in application architecture, 24
- Action bar
 - example in demo web application, 51
 - HTML template and page handler, 68
- Active Server Page
 - See ASP
- Active server page (ASP) script
 - definition of, 165
- ActiveX Business Object
 - See ABO
- ActiveX business object (ABO)
 - definition of, 165
- ActiveX DLL
 - definition of, 165
- Add as related document option
 - HTML templates, 99
- Add-Ins to Visual Basic, 20
- Administration page
 - example in demo web application, 42
 - HTML template and page handler, 68
- Allow user to keep session data option
 - description, 43
- ALTERNATE
 - HTML replacement tag, 125
- AppDictionary.bas, 66
 - customizing, 66
- Application library
 - definition of, 165

- Application service definition
 - definition of, 165
- Application services
 - definition of, 165
- Architecture
 - definition of, 165
- ASP, 65
 - role in application architecture, 25
 - WebAppF.asp, 65
 - WebAppFS.asp, 65

B

- BAS files
 - AppDictionary.bas, 66
 - customizing, 66
 - Globals.bas, 66
 - OFactory.bas, 66
 - TagProcessing.bas, 66
 - Utility.bas, 66
- BDT
 - changing field names
 - page handlers, 85
- BDT class
 - definition of, 166
- BDT controller class
 - definition of, 166
- BDT controller object
 - definition of, 166
- BDT modifier
 - definition of, 166
- BDT procedure
 - definition of, 166
- BestViewed.htm, 68
- BROWSE
 - HTML replacement tag, 125
- Browse command handler
 - definition of, 165

- Browse data cache
 - definition of, 165
- Browse dialog
 - definition of, 166
- Browse page
 - demo application
 - accessing, 53
- Browse pages
 - linking to other pages, 55
- Browse process
 - definition of, 166
- Browse.js
 - supplied JavaScript file, 73
- BROWSER
 - HTML replacement tag, 126
- Browsers
 - supported versions, 22
- BrowseTemplate.htm, 68
- Business data type (BDT)
 - definition of, 166
 - validating data, 144
- Business object
 - definition of, 167
- C**
- Calendar pop-up
 - example in demo web application, 50
 - HTML template, 68
- Cardinality
 - definition of, 167
- Cascading style sheet file
 - supplied
 - Styles.css, 67
- Change Password page
 - example in demo web application, 41
 - HTML template and page handler, 68
- CHECKBOX
 - HTML replacement tag, 126
- Child model
 - definition of, 167
- Client application
 - definition of, 167
- Client framework
 - definition of, 167
- Code
 - generated
 - protecting using implied user exits, 84
 - protecting using PRESERVE tag, 89
- Code block
 - definition of, 167
- Command block
 - definition of, 167
- Command handler
 - definition of, 167
- Command handler list
 - definition of, 167
- Command ID
 - definition of, 168
- Common.js
 - supplied JavaScript file, 73
- Complex redefine
 - definition of, 168
- Compression
 - definition of, 168
- Construct Spectrum
 - definition of, 168
 - documentation, 17
- Construct Spectrum Add-In
 - definition of, 168
- Construct Spectrum Administration subsystem
 - definition of, 168
 - role in application architecture, 26
- Construct Spectrum SDK
 - description, 20
 - documentation, 16
 - documentation and course information, 16
 - related products
 - HTML editors, 22
 - Visual Basic Add-Ins, 20
 - web server management, 22
 - supported web browsers, 22
- Construct Spectrum web application
 - architecture, 23
 - Microsoft Internet Information Server, 24
 - architecture of
 - Internet/intranet, 23
 - mainframe server, 26

- deployment
 - planning, 29
- designing, 30
- development
 - planning, 29
- development procedure, 28
- example, 35
- packaging and deploying, 155
 - before you begin, 156
 - manually, 156
 - using the Package and Deployment wizard
 - deploying the package, 160
- planning, 29
- role in application architecture, 24
- securing, 149
- Construct Spectrum web project
 - creating, 58
 - project directory, 60
- Control
 - changing, 106
 - changing captions, 113
 - derivation
 - HTML template, 106
 - radio button, 109
 - section
 - collapsible, 111
 - selection list, 107
 - text area
 - changing width, 113
 - text box
 - changing width, 113
- Conventions
 - typographical
 - used in this guide, 15
- Courses
 - related Natural Construct, 18
- CSS
 - See Cascading style sheet
- Customizing
 - BAS files, 66
 - HTML template before generating, 103
 - page handler, 84
 - security in Visual Basic
 - using Globals.bas, 152
 - using the object factory, 151

D

- Database record
 - definition of, 168
- DBID
 - definition of, 168
- Debug page
 - Application key, 45
 - example in demo web application, 44
 - HTML template and page handler, 68
 - Session key, 45
 - viewing cached errors, 146
- Debugging tools
 - definition of, 168
- Demo web application, 35
 - accessing, 36
- Dependent object
 - definition of, 169
- Deploying
 - Construct Spectrum web application, 155
 - web application
 - before you begin, 156
 - manually, 156
 - using the Package and Deployment wizard, 159
- Deployment
 - definition of, 169
 - planning, 29
- Develop
 - web application
 - procedure, 28
- Developing Web Applications
 - how to use guide, 13
 - layout, 11
- Development tools
 - Microsoft Management Console, 22
 - Microsoft Transaction Server, 22
 - Personal Web Server Manager, 22
- Dialog
 - definition of, 169
- Dispatch service
 - definition of, 169
 - See Spectrum dispatch service
- Dispatcher
 - definition of, 169

Documentation
 related Construct Spectrum, 17
 related Construct Spectrum SDK, 16
 related Natural Construct, 16–17

Domain
 definition of, 169

Double-byte character set (DBCS)
 definition of, 169

Download data
 definition of, 169

DRILL
 HTML replacement tag, 126

E

Encapsulation
 definition of, 169

Encryption
 definition of, 169

Entire Broker
 role in application architecture, 27

Entire Broker service settings
 definition of, 169

Entire Broker stub
 definition of, 169

ERROR
 HTML replacement tag, 127

ErrorHandler
 HTML template and page handler, 69

ERRORS
 HTML replacement tag, 127

Errors
 how displayed on web pages, 145

Event
 definition of, 169

Example web application, 35

Externally-creatable object
 definition of, 169

F

FIELD
 HTML replacement tag, 128

Field
 definition of, 170

Find button
 example in demo web application, 50

Flyout menu
 definition of, 170

FNR
 definition of, 170

Footer
 HTML template and page handler, 69

Foreign key
 definition of, 170

Form
 definition of, 170

Form section
 definition of, 170

Frames mode, 46
 Frameset HTML template and page handler, 69
 WebAppF.asp, 65
 WebAppFS.asp, 65

Framework components
 action bar, 51, 68
 Administration page, 42, 68
 ASP components, 65
 BAS files, 66
 Best Viewed page, 68
 Browse template, 68
 calendar pop-up, 50, 68
 cascading style sheets, 67
 Change Password page, 41, 68
 Debug page, 44, 68
 Error Handler page, 69
 Find button, 50
 footer, 69
 Frames, 46
 Frameset, 69
 Global.asa, 65
 header, 48, 69
 Home page, 38, 69
 in Project Explorer, 62
 JavaScript (.js) files, 73
 KeySelector, 69
 Layout template, 69
 Login page, 40, 69
 Logout page, 69
 Maint HTML template, 69
 message area, 52, 70
 navigation bar, 39, 70
 range options, 54

- section, 48
 - WebApp.cls, 65
 - WebAppF.asp, 65
 - WebAppFS.asp, 65
 - WebAppl.cls, 65
 - Framework HTML templates
 - role in application architecture, 25
 - Framework page handlers
 - role in application architecture, 25
 - Framework template
 - definition of, 170
- G**
- Generate
 - definition of, 170
 - Generated code
 - preserving using implied using exits, 84
 - protecting using PRESERVE tag, 89
 - Generated module
 - definition of, 170
 - Generation data cache
 - definition of, 170
 - Global.asa, 65
 - Globals.bas, 66
 - customizing, 66
 - customizing security, 152
 - Grid
 - definition of, 171
 - Grid control
 - definition of, 171
 - Group
 - definition of, 171
 - GUI
 - definition of, 171
 - GUI control override
 - definition of, 171
 - GUI keyword
 - HTML templates, 106
- H**
- Header
 - example in demo web application, 48
 - HTML template and page handler, 69
 - Home page
 - changing, 142
 - example in demo web application, 38
 - HTML template and page handler, 69
 - Hook
 - definition of, 171
 - Host
 - definition of, 171
 - HTML editors
 - Construct Spectrum SDK, 22
 - HTML fragment
 - definition of, 171
 - HTML Properties dialog
 - Links tab, 119
 - HTML Properties window
 - Links tab, 116
 - Section Type tab, 112
 - Select and Radio tab, 107
 - HTML replacement tag
 - parsing, 85
 - processing by page handler, 122
 - HTML tag
 - replacement
 - See also HTML replacement tag
 - HTML template
 - configuring, 98
 - customizing, 93
 - customizing before generation, 103
 - adding links, 114
 - changing captions, 113
 - changing column alignment, 118
 - changing controls, 106
 - changing header text, 120
 - changing section views, 110
 - changing width of text boxes and text areas, 113
 - creating links, 117–118, 120
 - deselecting fields for generation, 105, 117
 - specifying options in selection list, 107
 - specifying radio buttons, 109
 - definition of, 92, 171
 - examples in web page, 71
 - generating, 100
 - previewing, 101
 - Project Explorer, 102
 - role in application architecture, 25

HTML Template wizard
 configuring the template, 98
 confirming ABO details, 97
 definition of, 171
 invoking, 95
 selecting an ABO, 96
 using, 94, 100

HTML templates
 framework
 customizing, 93
 supplied with Construct Spectrum, 92

HTTP
 security, 41

http request
 definition of, 171

HTTPS, 41

I

IE
 See Microsoft Internet Explorer

IIS
 See Microsoft Internet Information
 Server

Implied user exit, 84

INDEX
 HTML replacement tag, 128

INFRAME
 HTML replacement tag, 128

INSTANCE
 HTML replacement tag, 129

Instantiation
 definition of, 171

Internationalization
 definition of, 172

Internet/Intranet
 role in application architecture, 23

J

JavaScript file
 supplied
 Browse.js, 73
 Common.js, 73
 Maint.js, 73

Job control language (JCL)
 definition of, 172

K

KeySelector
 and AppDictionary.bas, 66
 HTML template and page handler, 69

Keyword
 definition of, 172

L

Layout HTML template, 69

Level 1 data block
 definition of, 172

Level 1 data block optimization
 definition of, 172

Library image file (LIF)
 definition of, 172

LIF definitions module
 definition of, 172

Link
 from browse field to browse page, 120
 from browse to maintenance page, 118
 from maintenance field to another
 maintenance page, 117
 from maintenance field to browse page,
 114

LINK tag, 67

Linking
 browse pages to other pages, 55
 maintenance pages to other pages, 52

Links tab
 HTML Properties window, 116

Localization
 definition of, 172

LOGGEDOUT
 HTML replacement tag, 129

LOGIN
 HTML replacement tag, 129

Login page
 enabling in page handler, 82
 example in demo web application, 40
 HTML template and page handler, 69

- Logout page
 - HTML template, 69
- LOOKUP
 - HTML replacement tag, 129
- LOOKUP_VALUES
 - HTML replacement tag, 130
- Lookups
 - definition of, 172
- M**
- Mainframe server
 - role in application architecture, 26
- MAINT
 - HTML replacement tag, 130
- Maint HTML template, 69
- Maint.js
 - supplied JavaScript file, 73
- Maintenance dialog
 - definition of, 172
- Maintenance pages
 - linking to other pages, 52
- MDI child
 - definition of, 172
- MDI frame
 - definition of, 172
- MDI parent
 - definition of, 173
- MENU
 - HTML replacement tag, 130
- Menu
 - definition of, 173
 - flyout
 - definition of, 170
- Menu bar
 - definition of, 173
- Menus
 - fly-out
 - required browser versions, 22
- Message area
 - example in demo web application, 52
 - HTML template and page handler, 70
- Metadata
 - definition of, 173
- Method
 - definition of, 173
- Microsoft Internet Explorer, 22
- Microsoft Internet Information Server, 22
 - role in application architecture, 24
 - security, 153
- Microsoft Management Console
 - development tool, 22
 - setting session timeout, 41
- Microsoft Transaction Server
 - development tools, 22
 - role in application architecture, 24
- Model
 - definition of, 173
- MODIFY
 - HTML replacement tag, 130
- Module
 - definition of, 173
- MTS
 - See Microsoft Transaction Server
- Multi-level security
 - definition of, 173
- Multiple edit text area view, 111
- Multiple edit view, 111
- Multiple sort keys
 - example in demo web application, 54
- Multiple-document interface (MDI)
 - definition of, 174
- Multiple-valued field
 - HTML templates, 106
- N**
- Natural Construct nucleus
 - definition of, 174
- Natural Debugging facility
 - definition of, 174
- Natural subprogram
 - role in application architecture, 26
- Navigation bar
 - definition of, 174
 - example in demo web application, 39
 - HTML template and page handler, 70
- Navigator
 - See Netscape Navigator
- Netscape Navigator, 22
- Node
 - definition of, 174

NOFRAME

- HTML replacement tag, 130

Nonframes mode, 46

Nucleus

- definition of, 174

O**Object**

- definition of, 174

Object factory

- definition of, 174

- OFactory.bas, 66

- updating

 - using the Object Factory wizard, 139

- user exits, 142

- using to customize security, 151

- when to update, 138

Object Factory wizard

- definition of, 174

- generating an object factory, 141

- invoking, 139

Object library

- definition of, 174

OFactory.bas, 66

OLE

- definition of, 174

OLE automation server

- definition of, 174

ONLOAD_ADD

- HTML replacement tag, 130

Overflow condition

- definition of, 175

P**Package**

- definition of, 175

Package and Deployment wizard

- before using, 159

- using to create packages, 159

- using to deploy application, 160

PAGE

- HTML replacement tag, 131

Page handler

- coding custom browse actions, 89

- coding custom maintenance actions, 87

- coding validations, 87, 144

- customizing, 84

- definition of, 175

- framework

 - role in application architecture, 25

- ID, 81

- modifying the ParseTemplate function,

- 135

- parsing custom tags, 85

- processing HTML replacement tags,

- 122

- role in application architecture, 25

- role in assembling HTML templates, 92

- supplied user exits, 84

Page Handler wizard

- configuring, 81

- confirming ABO details, 80

- definition of, 175

- generating, 83

- invoking, 77

- selecting an ABO, 78

Parent model

- definition of, 175

PARENT_FIELDS

- HTML replacement tag, 131

PARENT_FORM

- HTML replacement tag, 131

Parse area

- definition of, 175

Periodic group

- HTML templates, 106

Personal Web Server Manager

- development tools, 22

Ping

- definition of, 175

Platform

- definition of, 175

POPUP

- HTML replacement tag, 131

Popup window

- displaying, 116

Predict

- verification rules

 - validating data, 145

PRESERVE tag, 89

Project
definition of, 175

Project group
definition of, 175

Project window
definition of, 175

Properties window
definition of, 175

Property
definition of, 175

R

RADIO
HTML replacement tag, 131

Range option
example in demo web application, 54

Regenerate/preserve status
definition of, 176

Remote call
definition of, 176

REPEAT
HTML replacement tag, 132

Replacement tag
ALTERNATE, 125
BROWSE, 125
BROWSER, 126
CHECKBOX, 126
creating, 93
creating HTML, 135
customizing for entire web application, 135
customizing for one page only, 135
definition of, 176
DRILL, 126
ERROR, 127
ERRORS, 127
FIELD, 128
INDEX, 128
INFRAME, 128
INSTANCE, 129
LOGGEDOUT, 129
LOGIN, 129
LOOKUP, 129
LOOKUP_VALUES, 130
MAINT, 130
MENU, 130
MODIFY, 130

NOFRAME, 130
ONLOAD_ADD, 130
PAGE, 131
PARENT_FIELDS, 131
PARENT_FORM, 131
POPUP, 131
processing by page handler, 122
RADIO, 131
REPEAT, 132
role in HTML template, 92
SECURITY, 132
SELECT, 133
SESSION, 133
SUBMIT, 133
syntax, 123
TITLE, 134
types, 123
UPDATEKF, 134
UPDATEPARENT, 134

Resource
definition of, 176

Run
definition of, 176

S

Section
example in demo web application, 48
collapsible, 48
view options, 49

Section Type tab
HTML Properties window, 112

Sections
collapsible, 111
HTML template
changing views, 110
multiple edit text area view, 111
multiple edit view, 111
single edit view, 110
single edit with report view, 110
view option
changing, 110

SECURITY
HTML replacement tag, 132

Security
customizing
using the object factory, 142

- customizing in Visual Basic
 - using Globals.bas, 152
 - using the object factory, 151
- HTTP, 41
- HTTPS, 41
- Login page, 40, 82
- supplied by IIS, 153
- supplied by Microsoft
 - key manager, 153
 - server certificate request, 153
 - SSL key pair, 153
- supplied with Construct Spectrum, 150
 - Construct Spectrum Administration subsystem, 150
 - login functionality, 150
 - security socket support, 150
- Security cache
 - definition of, 176
- Security service
 - role in application architecture, 27
- SELECT
 - HTML replacement tag, 133
- Select and Radio tab
 - HTML Properties window, 107
- Server
 - definition of, 176
- Server application
 - definition of, 176
- Service
 - definition of, 176
- Service exit
 - definition of, 176
- Service log
 - definition of, 176
- SESSION
 - HTML replacement tag, 133
- Session timeout
 - setting, 41
- Shutdown
 - definition of, 176
- Single edit view, 110
- Single edit view with report view, 110
- Software development kit (SDK)
 - definition of, 176
- Sort key
 - example in demo web application, 54
- Spectrum client/server application
 - definition of, 176
- Spectrum Control record
 - definition of, 176
- Spectrum Dispatch Client
 - role in application architecture, 24
- Spectrum Dispatch Client (SDC)
 - definition of, 177
- Spectrum dispatch service
 - definition of, 177
 - role in application architecture, 26
- Spectrum menu, 21
- Spectrum security service
 - definition of, 177
- Spectrum Service Manager
 - definition of, 177
- Spectrum service settings
 - definition of, 177
- Spectrum web application
 - definition of, 177
 - See Construct Spectrum web application
- Spectrum web framework
 - definition of, 177
- Spectrum web project
 - See Construct Spectrum web application
- Status bar
 - definition of, 177
- Steplib chain
 - definition of, 177
- Styles.css
 - supplied cascading style sheet (.css) file, 67
- Sub Main procedure
 - definition of, 177
- SUBMIT
 - HTML replacement tag, 133
- Subprogram proxy
 - definition of, 177
 - role in application architecture, 26
- Super model
 - definition of, 178

T

TagProcessing.bas, 66
 customizing, 66

TagProcessing.bas file
 modifying, 135

Target module
 definition of, 178

Target object
 definition of, 178

Target subprogram
 definition of, 178

Template parser
 definition of, 178

Text area
 changing width, 113

Text box
 changing width, 113

TITLE
 HTML replacement tag, 134

Toolbar
 definition of, 178

Toolbar button
 definition of, 178

Toolkit
 definition of, 178

Trace options
 definition of, 178

Type library
 definition of, 178

Typographical conventions
 used in this guide, 15

U

UPDATEKF
 HTML replacement tag, 134

UPDATEPARENT
 HTML replacement tag, 134

Upload data
 definition of, 178

User exits
 implied, 84
 supplied for page handlers, 84

Utility.bas, 66

V

Validating data
 coding in the page handler, 144
 in page handler, 87
 in the ABO, 144
 using business data types (BDTs), 144
 using Predict verification rules, 145
 validation types, 144

Variant
 definition of, 178

VB-Client-Server-Super-Model
 definition of, 178

Verification rule
 definition of, 179
 viewing, 108

Visual Basic
 Construct Spectrum SDK Add-Ins, 20
 Spectrum menu, 21

Visual Basic browse object
 definition of, 179

Visual Basic business object
 definition of, 179

Visual Basic maintenance object
 definition of, 179

W

Web application
 packaging and deploying, 155
 before you begin, 156
 manually, 156
 securing
 using Construct Spectrum security,
 150
 See Construct Spectrum web application

Web application ASP
 definition of, 179

- Web browsers
 - Microsoft Internet Explorer, 22
 - Netscape Navigator, 22
- Web class
 - definition of, 179
- Web server
 - supported, 22
- Web Super wizard
 - definition of, 179
- WebApp.cls, 65
- WebAppF.asp, 65
- WebAppFS.asp, 65
- Wildcard
 - definition of, 180
- Window
 - displaying popup, 116
- Wizards
 - HTML Template, 94
 - Object Factory, 139
 - Page Handler, 77

X

- XML extract
 - definition of, 180