

Developing Web Applications

Version 4.3

October 1999

CONSTRUCT SPECTRUM™ SDK

Manual Order Number: **SPV430-022IBW**

Copyright © SAGA SOFTWARE, Inc., 1999. All rights reserved.

SAGA, SAGA SOFTWARE, the SAGA logo, Free Your Information, the FYI logo, CRIS, Construct, Construct Spectrum, Construct Spectrum SDK, iXpress, Sagacertify, Sagagallery, Sagavista, and Your Fastest Route to Enterprise Integration are trademarks or registered trademarks of SAGA SOFTWARE, Inc. in the U.S. and/or other countries. Adabas, Adabas Delta Save Facility, Adabas Fastpath, Adabas SQL Server, Adabas Vista, Adaplex+, Bolero, Com-plete, Entire, Entire Access, Entire Net-work, EntireX, EntireX DCOM, Entire Broker, Entire Broker SDK, Entire Broker APPC, Entire SAF Gateway, Natural, Natural Architecture, Natural Elite, Natural New Dimension, Natural Lightstorm, Natural Vision, New Dimension, PAC, Predict, Software AG, and Super Natural are developed by Software AG of Darmstadt, Germany and are distributed in the U.S., Latin America, Canada, Israel and Japan exclusively through SAGA SOFTWARE, Inc. and its subsidiaries and distributors. Adabas and Natural are registered trademarks of Software AG of Darmstadt, Germany. Except for Adabas and Natural, these products developed by Software AG of Darmstadt, Germany are either registered trademarks or trademarks of SAGA SOFTWARE, Inc., in the U.S. and/or other countries.

Other company or product names mentioned are used for informational purposes only and may be trademarks or servicemarks of their respective owners.

TABLE OF CONTENTS

PREFACE

Prerequisite Knowledge	12
How to Use this Guide	13
If You are Creating Web Components Only	13
If You are Creating All Components of a Web Application	14
Conventions Used in this Guide	15
Related Documentation	17
Construct Spectrum SDK	17
Construct Spectrum	18
Natural Construct	18

1. INTRODUCTION

What is Construct Spectrum?	20
Construct Spectrum and Related Tools	20
Construct Spectrum Add-Ins to Visual Basic	20
HTML Editor	22
Web Browsers	22
Web Server Management	23
Architecture of a Construct Spectrum Web Application	24
Mainframe Server	25
Internet Information Server (IIS)	27
Internet/Intranet	30
The Development Process	31
Step 1 — Planning and Designing Your Web Application	31
Planning for Deployment and Operation	32
Planning for Development	32
Designing the Web Application	33
Step 2 — Setting Up Your Application Environment	33
Step 3 — Generating Natural Components	34
Step 4 — Creating an ABO Project	34

Step 5 — Generating ActiveX Business Objects	34
Step 6 — Creating a Spectrum Web Project	35
Step 7 — Generating Web Components	35
Step 8 — Customizing Your Application	36
Step 9 — Testing and Debugging Your Application	36
Step 10 — Deploying Your Application	36

2. FEATURES OF THE DEMO WEB APPLICATION

Accessing the Demo Web Application	38
Features of the Home Page and Navigation Bar	39
Home Page	39
Navigation Bar	41
Login Page	42
Administration Page	44
Debug Page	45
Change Password Page	48
Frames	50
Features of a Maintenance Page	52
Header	54
Sections	54
Collapsible Sections	55
View Options	55
Find Buttons	56
Calendar Pop-Up	57
Action Bar	57
Message Area	58
Features of a Browse Page	59
KeySelector Template	61
Multiple Sort Keys	61
Range Options	61
Go and More Buttons	61
Browse Rows	61

3. CREATING A WEB PROJECT	
Using the Spectrum Web Project Wizard	64
4. FRAMEWORK COMPONENTS SUPPLIED WITH CONSTRUCT SPECTRUM	
Introduction	72
Active Server Components and the WebApp.cls.	73
BAS Files	74
Customizing BAS Files	74
AppDictionary.bas	75
Globals.bas	75
TagProcessing.bas	75
Cascading Style Sheets	76
Framework HTML Templates and Page Handlers	77
Examples of HTML Templates in Web Pages	80
JavaScript Files	82
5. GENERATING AND CUSTOMIZING PAGE HANDLERS	
Using the Page Handler Wizard	86
Invoking the Wizard.	86
Selecting an ABO	88
Confirming ABO Details	90
Configuring the Page Handler.	92
Generating the Page Handler	95
Customizing Page Handlers	97
Protecting Generated Code	97
Implied User Exits	97
User Exits in Page Handlers	97
User Exits in Maintenance Page Handlers.	97
ICSTPageHandler_Content.CustomContentIDs	98
ICSTPageHandler_BDTOverrides.	98
ParseTemplate.CustomTags	98
PerformAction.OtherResets	99
PerformAction.CustomUpdateActions	99
PerformAction.UpdateForeignKeys.	99
PerformAction.ClientValidations.	100

PerformAction.CustomActions	100
RetrieveFromSession.CustomState and StoreToSession.CustomState	101
UpdateData.CustomUpdates	101
User Exits in Browse Page Handlers	101
ICSTPageHandler_Process.CustomActions	101
ICSTPageHandler_BDTOverrides	102
RetrieveFromSession.CustomState and StoreToSession.CustomState	102
ParseTemplate.CustomTags	102
ICSTPageHandler_Content.CustomContentIDs	103

6. GENERATING AND CUSTOMIZING HTML TEMPLATES

Introduction	106
Framework HTML Templates	106
Replacement HTML Tags	106
Customizing HTML Templates	107
Before Generating the Template	107
After Generating the Template	107
Using Replacement Tags	107
Using Framework Components	107
Using the HTML Template Wizard	109
Invoking the Wizard	109
Selecting an ABO	111
Confirming ABO Details	113
Configuring the HTML Template	114
Generating the HTML Template	117
After Generation is Complete	121
Customizing HTML Before Generation	123
Customizing Maintenance Pages	125
Deselecting Fields for Generation	125
Changing the Type of Control for a Field	125
How Controls are Derived	126
Controls in Construct Spectrum Web Applications	126
Changing the Control for a Field	127
Specifying or Modifying Options in a Selection List	127
Specifying or Modifying Options in a Radio Button Group	130
Changing View Options for Sections	131

Single Edit View	132
Single Edit View with Report View Option	132
Multiple Edit View	133
Multiple Edit Text Area View	133
Collapsible Sections	134
Changing View Options	134
Changing the Width of a Text Box or Text Area	136
Changing the Control's Caption	136
Creating a Link to a Browse Page	136
Creating a Link to a Browse Page	138
Customizing Browse Pages	140
Deselecting Fields for Generation	140
Changing the Alignment of a Column in a Browse Table	140
Adding a Link to a Maintenance Page	140
Creating a Link to a Maintenance Page	141
Changing Header Text	143

7. CONSTRUCT SPECTRUM REPLACEMENT HTML TAGS

How Page Handlers Process Tags	146
Syntax of Replacement Tags	147
Types of Replacement Tags	148
Simple	148
Conditional	148
Repeating	148
Complex	149
Replacement Tags Supplied with Construct Spectrum	150
ALTERNATE	150
BROWSE	150
BROWSER	151
CHECKBOX	151
ERROR	152
ERRORS	152
FIELD	153
INDEX	153
INFRAME	154
INSTANCE	154

LOGGEDIN	154
LOGGEDOUT.....	155
LOOKUP.....	155
MAINT	155
PAGE.....	156
RADIO.....	156
REPEAT	157
SECURITY	157
SELECT	158
SUBMIT	158
TITLE	159
Defining Custom HTML Replacement Tags.....	160
Modifying TagProcessing.bas	160
Modifying the Page Handler	161
8. UPDATING AND CUSTOMIZING THE OBJECT FACTORY	
Introduction	164
Using the Object Factory Wizard	165
User Exits in the Object Factory.....	170
DefaultPage.SetDefault	170
Security User Exits	170
9. VALIDATING YOUR DATA	
Types of Validations Used in Web Applications.....	172
BDTs	172
Validations in the ABO	172
Validations in the Page Handler	173
How Errors are Displayed in Web Pages	174
How Errors Are Displayed in Internet Explorer	174
How Errors are Displayed in Navigator	175
The Debug Page	177
10. SECURING YOUR APPLICATION	
Security Supplied with Construct Spectrum	180
Spectrum Security	180
Security Sockets	180

Login Functionality	180
Customizing Security	181
Coding User Exits in the Object Factory	181
IsPermitted.Override	181
ValidateUser.Override	182
IsPermittedCustomTags	182
Globals.bas	182
Security Supplied by Microsoft Internet Information Server	183
The Key Manager	183
The Server Certificate	183
The SSL Key Pair	183
11. DEPLOYING YOUR WEB APPLICATION	
Before Deploying the Application	186
Manual Deployment	187
Creating a New Package for the Web DLL	187
Creating a Virtual Directory	188
Creating a Starting Point for the Application	188
Modifying Application Settings	188
Using the Package and Deployment Wizard	190
Before Using the Package and Deployment Wizard	190
Creating the Distributable Package	190
Deploying the Package	191
INDEX	193

PREFACE

Welcome to *Developing 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 out of the guide and find other sources of information about creating Construct Spectrum web applications.

The following topics are covered:

- **Prerequisite Knowledge**, page 12
- **How to Use this Guide**, page 13
- **Conventions Used in this Guide**, page 15
- **Related Documentation**, page 17

Prerequisite Knowledge

Developing Web Applications does not provide information about the following topics. We assume that you are either familiar with these 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[®]

How to Use this Guide

Developing Web Applications explains how to create and customize the web components (HTML templates, page handlers, and object factory entries) of an application. It also explains how to secure and deploy your web application. However, before you can start generating and customizing web components, 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 to you. The web components you generate use the information in these objects to present and modify business data.

The following paths through the documentation explain two scenarios: if you are creating web components only and if you are creating a complete web application.

If You are Creating Web Components Only

If the Natural and ActiveX components of your application are already available to you, *Developing Web Applications* provides almost all of the information you need to create and customize web components and to deploy your web application. We also recommend that you read the following chapters in the *Construct Spectrum Programmer's Guide*:

- Chapter 3, **Features of the ABO and Web Wizards** for information about setting configuration options for the wizards, using Spectrum's client-side cache, and modifying code frames.
- Chapter 7, **Using Business Data Types** for information about using BDTs in your web applications.
- Appendix A, **Glossary of Terms** for definitions of common terms used in the Construct Spectrum documentation.

If You are Creating All Components of a Web Application

If you wish to create a complete web application, including Natural modules and ActiveX business objects, first refer to the *Construct Spectrum Programmer's Guide*. We recommend that you read the following chapters:

- Chapter 1, **Introduction** for overviews of the product, the application architecture, and the development process.
- Chapter 2, **Setting Up Your Application Environment on the Mainframe** for detailed information about how to define domains and security options to control what data users of your application will access on the mainframe.
- Chapter 3, **Features of the ABO and Web Wizards** for information about setting configuration options for the wizards, using Spectrum's client-side cache, and modifying code frames.
- Chapter 4, **Using the Business-Object Super Model** for detailed information about how to use this model wizard to generate the Natural components of your application.
- Chapter 5, **Using ActiveX Business Objects** for detailed information about creating ABOs and an ABO project to contain them using the wizards supplied with Construct Spectrum.
- Appendix A, **Glossary of Terms** for definitions of common terms used in the Construct Spectrum documentation.

When you are ready to create the web components of your application, turn to *Developing Web Applications* for detailed information about generating HTML templates, page handlers, and object factory entries. This guide also explains how to customize, debug, secure, and deploy your web application.

As you customize and regenerate your application components, you will find these chapters in the *Construct Spectrum Programmer's Guide* useful:

- Chapter 6, **Using the Subprogram Proxy Model**
- Chapter 7, **Using Business Data Types**

Conventions Used in this Guide

This guide uses the following typographical conventions:

Example	Description
Introduction	Bold text in cross references indicates chapter and section titles.
“A”	Quotation marks indicate values you must enter.
Browse model, GotFocus, Enter	Mixed case text indicates: <ul style="list-style-type: none"> • The names of Natural Construct and Construct Spectrum editors, fields, files, functions, models, panels, parameters, subsystems, variables, and dialogs. • The names of Visual Basic classes, constants, controls, dialogs, events, files, menus, methods, properties, and variables. • The names of keys.
Alt+F1	A plus sign (+) between two key names indicates that you must press the keys together to invoke a function. For example, Ctrl+S means hold down the Ctrl key while pressing the S 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 Administrator’s Guide, variable name</i>	Italicized text indicates: <ul style="list-style-type: none"> • Book titles. • Placeholders for information you must supply.

Example	Description (continued)
<code>[variable]</code>	In syntax and code examples, values within square brackets indicate optional items.
<code>{WHILE UNTIL}</code>	In syntax examples, values within brace brackets indicate a choice between two or more items; each item is separated by a vertical bar ().

Related Documentation

The documentation sets for Construct Spectrum, Construct Spectrum SDK, and Natural Construct consist of the following manuals:

Construct Spectrum SDK

- *Construct Spectrum Programmer's Guide*
This guide 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.
- *Developing Web Applications*
This guide is for developers creating the web components of applications. It explains how to use the Construct Spectrum wizards in Visual Basic to generate HTML templates, page handlers, and object factory entries. It also contains detailed information about customizing, deploying, and securing web applications.
- *Construct Spectrum Reference Manual*
This manual is for application developers and administrators who need quick access to information about Construct Spectrum application programming interfaces (APIs) and utilities.
- *Construct Spectrum Messages*
This manual is for application developers, application administrators, and system administrators who wish to investigate messages returned by Construct Spectrum run-time and SDK components.
- *Developing Client/Server Applications*
This guide is for developers creating client components for applications that will run in the Natural mainframe environment (server) and a Windows environment (client).

Construct Spectrum

- *Construct Spectrum and SDK Client Installation*
This manual explains how to install and set up the Construct Spectrum run-time and SDK components on the client.
- *Construct Spectrum and SDK Mainframe Installation*
This manual explains how to install and set up the Construct Spectrum run-time and SDK components on the mainframe.
- *Construct Spectrum Administrator's Guide*
This guide is for administrators who wish use the Construct Spectrum Administration subsystem to set up and manage Construct Spectrum applications.

Natural Construct

- *Natural Construct Installation and Operations Manual for Mainframes*
This manual 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 User's Manual*
This manual explains how to use the Natural Construct models to generate applications that will run in a mainframe environment.
- *Natural Construct Administration and Modeling User's Manual*
This manual explains how to use the Administration subsystem of Natural Construct and how to create new models.
- *Natural Construct Help Text User's Manual*
This manual explains how to create online help for applications that run on server platforms.

INTRODUCTION

This chapter explains the components of Construct Spectrum and the architecture of the web applications you can create with the software development kit (SDK). An overview of the steps involved in developing a web application prepares you for the detailed procedures in the chapters that follow.

The following topics are covered:

- **What is Construct Spectrum?**, page 20
- **Architecture of a Construct Spectrum Web Application**, page 24
- **The Development Process**, page 31

What is Construct Spectrum?

Construct Spectrum and the software development kit (SDK) comprise a set of middleware and framework components, as well as integrated tools that use the specifications you supply to generate all the components of a web application. These components include Natural modules that perform maintenance and browse functions on database records, ActiveX business objects that facilitate communication between client and server, and HTML pages that run in a browser to present business data to users.

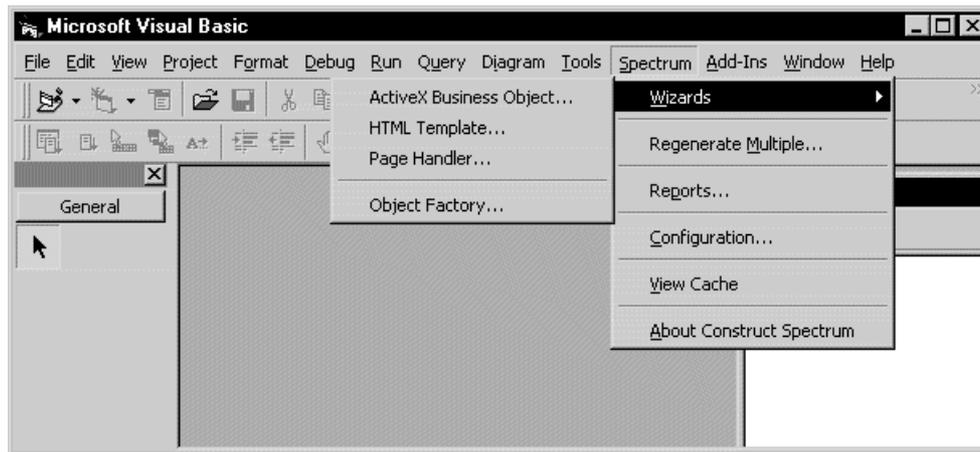
This guide provides information about generating and customizing the web components of your application: page handlers, HTML templates, and object factory entries. For information about creating Natural modules and ActiveX business objects, see the *Construct Spectrum Programmer's Guide*.

Construct Spectrum and Related Tools

In the process of creating the web components of an application, you will typically work with several development and design tools.

Construct Spectrum Add-Ins to Visual Basic

Construct Spectrum's wizards and utilities for developing web components are integrated into the Visual Basic development environment. The Construct Spectrum Add-Ins to Visual Basic are evident when you add a new Construct Spectrum web or ABO project from the **New Project** dialog or **Add Project** dialog. Also, when you open Visual Basic after installing Construct Spectrum, you will notice the **Spectrum** menu:



Spectrum Menu in Visual Basic

Using the **Spectrum** menu, you can:

- Use wizards for generating web components: ABOs, HTML templates, page handlers, and object factory entries.
- Regenerate individual or multiple modules
- Edit and regenerate a module using the original wizard
- View generation reports and compare code
- View and set options in the Spectrum cache
- View and modify environmental options in the Configuration editor

For information about using the Regenerate, Edit, Reports, Configuration, and View Cache commands, see **Features of the ABO and Web Wizards**, page 59, *Construct Spectrum Programmer's Guide*.

HTML Editor

In addition to Visual Basic's development environment, you can also use the HTML editor of your choice to customize your HTML templates. We have tested the following editors:

- Microsoft® Visual InterDev®
- Microsoft FrontPage® 98 (or higher)
- Homesite®

Tip: We recommend editing your pages by changing the HTML directly, rather than dragging and dropping elements in your editor's WYSIWYG view. Construct Spectrum applications use special tags for replacing content dynamically and laying out page elements. In a WYSIWYG editor, you cannot see these tags and, consequently, it is very easy to get them out of order.

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

Web Browsers

You can run and test your Construct Spectrum web application using Microsoft® Internet Explorer® 4.0 (or later) or Netscape Navigator® 4.0 (or later). We recommend version 5.0 of Internet Explorer because it provides improved HTML rendering and the ability to bookmark web pages in Frames mode.

Because Internet Explorer supports a greater subset of DHTML tags than Navigator, not all of the functionality that it is possible to incorporate in your web applications is available to users of Navigator.

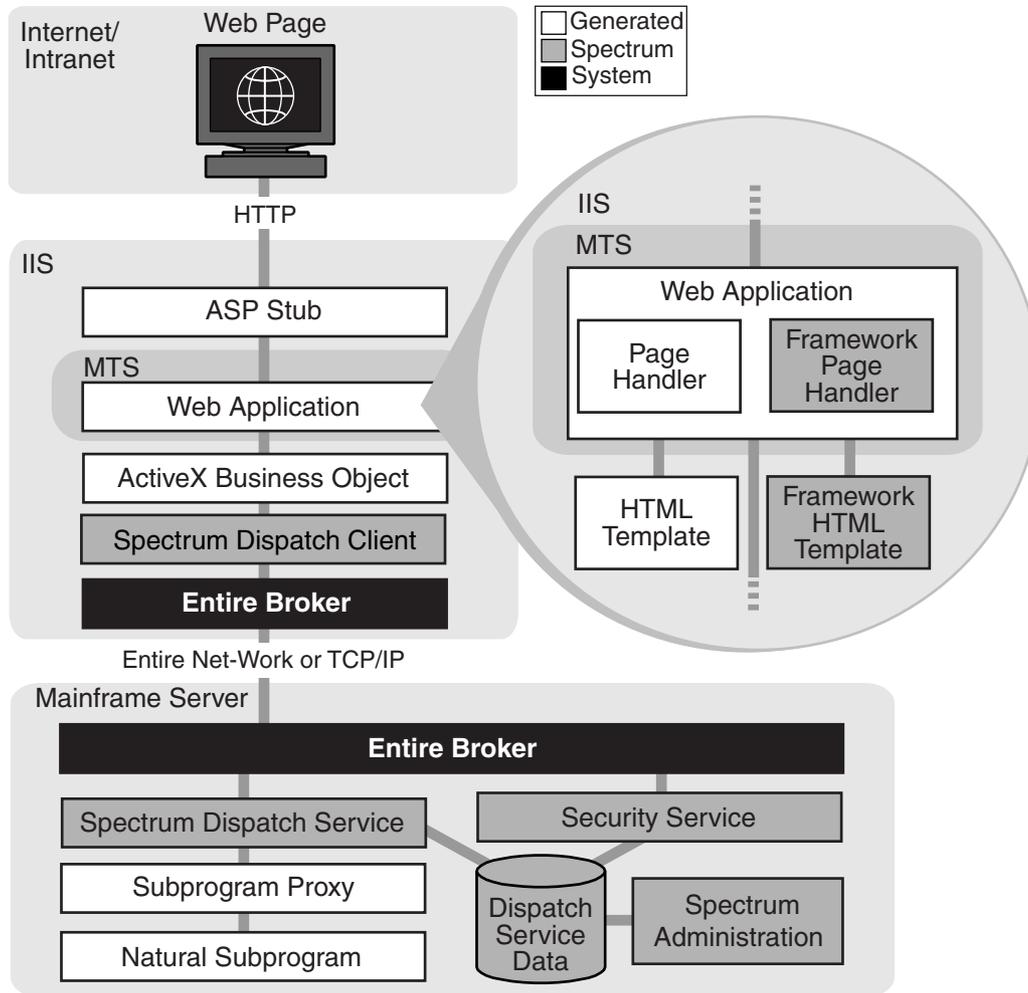
Construct Spectrum applications automatically provide alternate functionality to accommodate different browsers. For example, in Internet Explorer, errors on fields are highlighted. When viewed in Navigator, errors are indicated by a graphic.

Web Server Management

You will also use Microsoft Management Console® (MMC) to work with the Microsoft Internet Information Server® (IIS) on Windows NT and to administer the Microsoft Transaction Server® (MTS), which runs the deployed Spectrum web application. If you are developing your web applications on Windows, you will make use of the Personal Web Server® Manager.

For more information about IIS, MTS, MMC, and Personal Web Server, see the documentation that Microsoft supplies for these products.

Architecture of a Construct Spectrum Web Application



Architecture of a Construct Spectrum Web Application

The following sections explain the components shown in the illustration according to the platforms on which the components run: mainframe server, IIS, and internet or intranet.

Mainframe Server

Component	Description
Natural subprogram	<p>Natural subprograms perform maintenance and browse functions on the mainframe server. Construct Spectrum web applications are capable of communicating with both subprograms that are created for the web application and existing Natural subprograms.</p> <p>For more information about creating subprograms for a web application, see Using the Business-Object Super Model, page 87, <i>Natural Construct Programmer's Guide</i>.</p>
Subprogram proxy	<p>A subprogram proxy acts as a bridge between a specific subprogram and the Spectrum dispatch service. It performs a number of functions, including translating parameter data into a format that can be transmitted between web page and mainframe server, issuing CALLNATs to subprograms, and validating the format and length of data received from the client.</p> <p>For more information, see Using the Subprogram Proxy Model, page 129, <i>Construct Spectrum Programmer's Guide</i>.</p>

Component	Description (continued)
Spectrum dispatch service	<p>Spectrum dispatch services ensure that the current user is allowed to perform the requested function. Once the service has performed user authentication, it activates the correct Natural subprogram to handle the request. After the target subprogram completes, the results are transferred back to the client. Depending on user options, the service may also be required to compress and decompress and/or encrypt and decrypt messages.</p> <p>For more information, see Defining and Managing Construct Spectrum Services, page 47, <i>Construct Spectrum Administrator's Guide</i>.</p>
Dispatch service data	<p>The information defined and maintained in the Spectrum Administration subsystem is accessed by Spectrum dispatch services anywhere on the network by way of Entire Broker.</p>
Spectrum administration	<p>This mainframe subsystem allows system administrators, application administrators, and application developers to set up and manage system and application environments.</p> <p>For more information, see <i>Construct Spectrum Administrator's Guide</i>.</p>
Security service	<p>A Spectrum security service 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. Its independence allows the security service to process, in one central location, the requests of several Spectrum dispatch services, which may be located on nodes throughout the network.</p> <p>For more information about security services and security settings, see <i>Construct Spectrum Administrator's Guide</i>.</p>
Entire Broker	<p>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.</p>

Internet Information Server (IIS)

Web applications created with Construct Spectrum work with IIS.

Component	Description
Entire Broker	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>This Component Object Model (COM) middleware component enables web applications to read from, and write to, variables in a Natural parameter data area (PDA) and to issue CALLNAT statements to Natural subprograms. Its main functions are simulating PDAs and CALLNATs, encapsulating Entire Broker calls, and controlling database transactions. As the client counterpart of Spectrum dispatch services, it is also responsible for such things as data marshaling, encryption, compression, error-handling and all Entire Broker communication.</p> <p>For more information, see Understanding The Spectrum Dispatch Client, page 243, <i>Construct Spectrum Programmer's Guide</i>.</p>
ActiveX Business Object	<p>Each back-end business object is represented on the web server as an ActiveX business object. This object encapsulates all of the communication with the Spectrum Dispatch Client, making it efficient to invoke Natural services from the client.</p> <p>For more information, see Using ActiveX Business Objects, page 107, <i>Construct Spectrum Programmer's Guide</i>.</p>

Component	Description (continued)
MTS	The Microsoft Transaction Server (MTS) provides a controlled environment from which to run COM web components. It improves the scalability of applications, allowing them to expand to meet future needs. It also provides a centralized facility for shutting down processes and updating the ActiveX dynamic-link libraries (DLLs) that contain web applications. MTS also makes the functionality of Active Server Pages (ASPs) available to your web applications.
Web application	A Construct Spectrum web application consists of framework components supplied with all Construct Spectrum web projects and components that you generate using Construct Spectrum wizards. Generated components are HTML templates, page handlers, and object factory entries. Framework components include standard HTML templates, helper routines, and the ASP files required to run your application.
Page handler	<p>A page handler is a Visual Basic class used to manage web requests and responses associated with HTML content that is supplied programmatically.</p> <p>For more information, see Generating and Customizing Page Handlers, page 85.</p>

Component	Description (continued)
HTML template	<p data-bbox="635 318 1366 471">An HTML template consists mainly of HTML and some JavaScript™, which present a web page or a component of a web page in the browser. A number of Construct Spectrum components execute on the web server to build a web page by assembling HTML templates.</p> <p data-bbox="635 485 1366 722">Templates may contain placeholders that can be replaced with active content, such as database information. One of the Construct Spectrum framework components is a template parser that parses a designated starting template and resolves references to sub-templates and active content. Templates are processed until all placeholders have been resolved. The resulting HTML is sent to the browser as an HTTP response.</p> <p data-bbox="635 736 1366 795">For more information, see Generating and Customizing HTML Templates, page 105.</p>
Framework page handler and framework HTML template	<p data-bbox="635 820 1366 999">Framework components include HTML templates and page handlers for the navigation bar, header, footer, login page, etc. that are included with every Construct Spectrum web application (and which you can customize for your application). Utilities and helper routines are also supplied in the framework.</p> <p data-bbox="635 1014 1366 1076">For more information, see Framework Components Supplied with Construct Spectrum, page 71.</p>
ASP Stub	<p data-bbox="635 1102 1366 1250">An Active Server Page (ASP) script is 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 data-bbox="635 1265 1366 1333">For more information, see Active Server Components and the WebApp.cls, page 73.</p>

Internet/Intranet

Construct Spectrum web applications support Internet Explorer and Netscape Navigator browsers at version 4.0 and later. Internet Explorer 5.0 is recommended because it provides improved HTML rendering and the ability to bookmark web pages in Frames mode.

The Development Process

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

- 1 Plan and design your web application
- 2 Set up your application environment
- 3 Generate Natural components
- 4 Create an ABO project (or open an existing ABO project)
- 5 Generate ActiveX business objects (or use existing ABOs)
- 6 Add a web project to your project group
- 7 Generate the following web components:
 - page handlers
 - HTML templates
 - object factory entries
- 8 Customize your application
- 9 Test and debug your application
- 10 Deploy your application

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

Note: This guide provides information about steps 1 and 6-10. Detailed information about steps 2-5 are contained in Construct Spectrum Programmer's Guide.

Step 1 — Planning and Designing Your Web Application

Planning your web application is important because there are so many factors that must be decided if the project is to be implemented efficiently, without major revisions to accommodate changes. The following sections point out some of the issues.

Planning for 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.
- What security model is appropriate for your application? For more information, see **Securing Your Application**, page 179.
- How many users do you expect in your audience? Can your web server handle that many? How will mainframe and web servers perform as the number of users grows?
- How long will it take to download graphics and other objects? Smaller downloadable files will be appreciated by users, especially if your application will be accessed on the internet.
- What web browsers are users likely to employ? If you are building an application to be deployed for an internal audience, the type and version of the browser may be standardized for users.

Planning for Development

Will web applications be developed by a team of two or twenty? 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:

- Web designers will appreciate a separate internet server to test their pages.
- Standardize on one HTML editor for everyone to use to ensure that code is formatted consistently.

Designing the Web Application

Much has been written about designing web sites and pages that are consistent, usable, and easy to navigate. Here are some considerations specific to designing Construct Spectrum web applications:

- Construct Spectrum’s framework HTML templates, cascading style sheets, and code frames are built to be customized. They provide a starting point for building a unique and consistent corporate style for your applications. For example, you could modify the `Layout Template.htm` framework file supplied with Construct Spectrum so that your corporate logo appears in the same location on each web page that you generate.
- Construct Spectrum’s HTML tags also provide flexibility in designing your web pages because you can embed tags that are replaced with content at runtime.

Step 2 — Setting Up Your Application Environment

Before a new web application can be created, some measures have to be taken to ensure that it accesses the correct Natural business objects and that users can access the application. This involves the following tasks:

- Setting up Predict definitions. Construct Spectrum works with Predict, a data dictionary and repository that manages metadata about the information contained in the database that your applications use. Predict’s “views” of data and the relationships between data help you define the business objects your applications access and maintain. Predict’s verification rules and keywords are used to validate and format data, and its field definitions automatically select controls for your applications. You can do a lot with Predict to define the defaults Construct Spectrum uses to generate your applications.
- Defining a steplib chain and domain in the Construct Spectrum Administration Subsystem and associating the steplib chain with the domain.
- Setting up security privileges for the domain. This involves setting up users and groups and granting them access to the domain, its objects and methods.

For more information, see **Setting up Your Application Environment on the Mainframe**, page 45, *Construct Spectrum Programmer’s Guide*.

Step 3 — Generating 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. The Business-Object super model allows you to generate these components for up to 12 business object “packages” at a time.

For more information, see **Using the Business-Object Super Model**, page 87, *Construct Spectrum Programmer’s Guide*.

Step 4 — Creating 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). During development, this makes it easy to add a reference to several objects at one time. Also, if these business objects are shared among web and client/server applications, combining multiple objects within a single DLL simplifies the deployment of these objects.

Construct Spectrum provides the Spectrum ABO wizard as part of the Visual Basic development environment for creating ABO projects, which you can later compile into DLLs.

For more information, see **Using ActiveX Business Objects**, page 107, *Construct Spectrum Programmer’s Guide*.

Step 5 — Generating ActiveX Business Objects

Each Natural business object is represented on the web server by an ActiveX business object. These COM objects make it very convenient for client components to interface with server objects using standard interfaces.

Construct Spectrum provides the ABO wizard to define and generate ABOs. For more information, see **Using ActiveX Business Objects**, page 107, *Construct Spectrum Programmer’s Guide*.

Step 6 — Creating 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 Spectrum Web wizard in Visual Basic, with which you can generate the web project. When the generation is complete, several framework components are included in the project.

For more information, see **Creating a Web Project**, page 63.

Step 7 — Generating Web Components

For each ABO, you can generate the following components:

- **Page handlers**
You will need a page handler for each HTML template to process the web page and replace tags. Use Construct Spectrum's Page Handler wizard in Visual Basic to create the page handlers.

For more information, see **Generating and Customizing Page Handlers**, page 85.

- **HTML templates**
Using the HTML Template wizard, you can generate the HTML files for each page in your application.

For more information, see **Generating and Customizing HTML Templates**, page 105.

- **Object Factory entries**
Construct Spectrum web applications make use of a Visual Basic module called an object factory. The object factory encapsulates all the business objects used by an application. When you add or modify an ABO or page handler, you can use the Object Factory wizard in Visual Basic to update the object factory.

For more information, see **Updating and Customizing the Object Factory**, page 163.

Step 8 — Customizing Your Application

Once you have generated 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. This iterative process may require you to regenerate modules using Construct Spectrum's regeneration facilities.

For more information, see:

- **Generating and Customizing Page Handlers**, page 85.
- **Generating and Customizing HTML Templates**, page 105.
- **Framework Components Supplied with Construct Spectrum**, page 71.

Step 9 — Testing and Debugging Your Application

In the process of generating and customizing your application, you will wish to run it in Internet Explorer or Navigator to test the functionality. You can use Visual Basic's debugging features to resolve errors, as well as the features Construct Spectrum supplies for trouble-shooting communication problems between the client and server.

For information about using the framework Debug page, see **The Debug Page**, page 177.

Step 10 — Deploying Your Application

When your application is complete, you can create dynamic link library (DLL) files for your ABO and Spectrum web projects, package the application files, install them on an internet server, and configure application settings.

For more information, see **Deploying Your Web Application**, page 185.

FEATURES OF THE DEMO WEB APPLICATION

This chapter presents a selective tour of an example web application that was created with Construct Spectrum to be installed with the product. This application is designed to demonstrate the features of a simple Construct Spectrum web application. You can explore this application to become familiar with the components and functionality that you can use in your own applications.

The Demo web application is a simple Order Entry application that enables users to browse and maintain four business objects: Order, Customer, Product, and Warehouse. The application retrieves and updates data on the mainframe using Predict files installed in a demo library.

This chapter describes many of the features and functions of the Demo web application, using the Order Maintenance and Customer Browse pages to illustrate. Some of the features are provided by default with every application developed with Construct Spectrum; others are provided based on the Predict setup of your application; and others are the result of adding custom code.

The following topics are covered:

- **Accessing the Demo Web Application**, page 38
- **Features of the Home Page and Navigation Bar**, page 39
- **Features of a Maintenance Page**, page 52
- **Features of a Browse Page**, page 59

Accessing the Demo Web Application

The Demo web application is installed with the web component of Construct Spectrum SDK. By default, the installation process places it in the publishing path of your web server. (For example, if you are using Personal Web Server, the demo application is installed into `Inetpub\wwwroot\WebOrderEntry`.)

As a result of the installation and configuration processes, a Spectrum dispatch service should be available to you on the web server. For more information about installing and configuring Construct Spectrum SDK, see *Construct Spectrum and SDK Client Installation*.

Note: If you are using Entire Net-Work, make sure that an Entire Net-Work kernel is running on your PC before you try to open the Demo application.

- To open the Demo web application:
- 1 Open Internet Explorer or Navigator.
 - 2 Provide the following URL:

`http:\hostname\WebOrderEntry\CstOrderEntryDemo\WebApp.asp`

where *hostname* is the name of your PC or “localhost”.

The Home page of the Demo application appears.

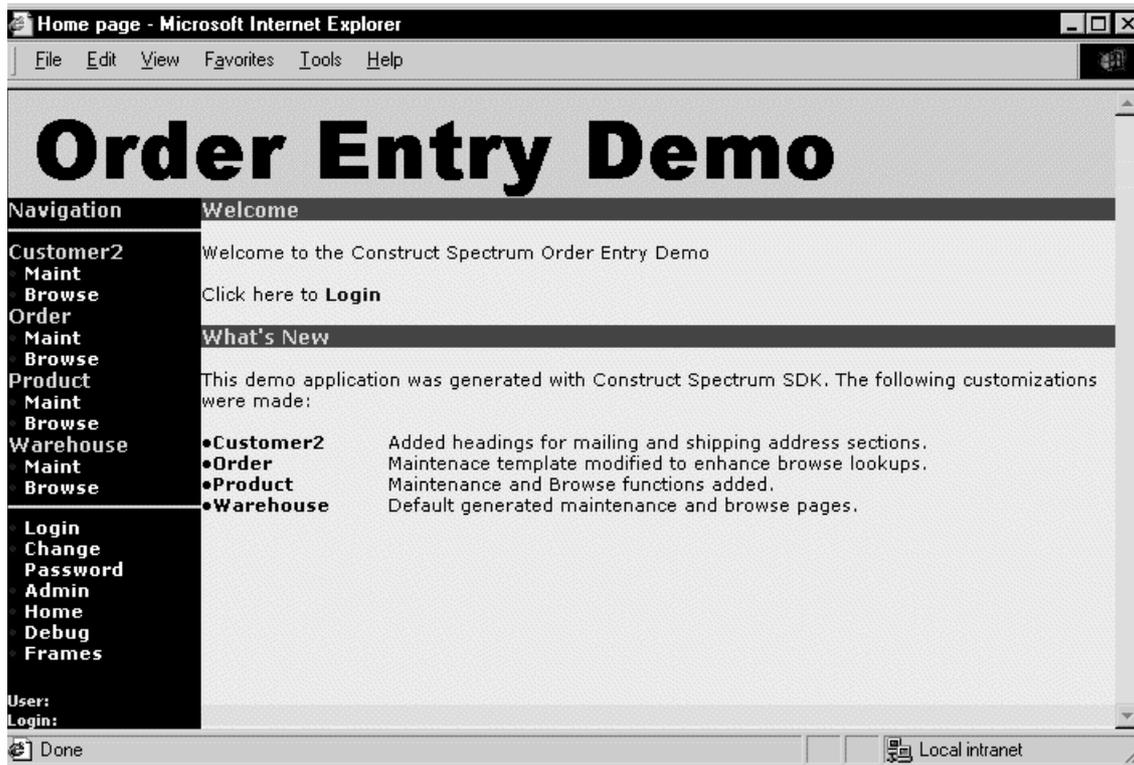
The next section explains the features of the Home page and navigation bar, as well as how to log in to the Demo application and use the Administration page to view your dispatch services.

Features of the Home Page and Navigation Bar

The Home page is the starting point for the application. The navigation bar is present in all web pages to give users access to business objects and framework components: Login page, Administration page, Change Password page, Debug page, and Frames mode.

Home Page

When you open the Demo application, the Home page for the Order Entry application appears in the browser. The illustrations in this chapter show the pages as seen in Internet Explorer.



Order Entry Application — Home Page

The Home page is a framework component supplied with Construct Spectrum web projects. You can use it to provide introductory information and links. To customize the Home page, work with the page handler `Home.cls` and `Home.htm` that Construct Spectrum places in the web application directory.

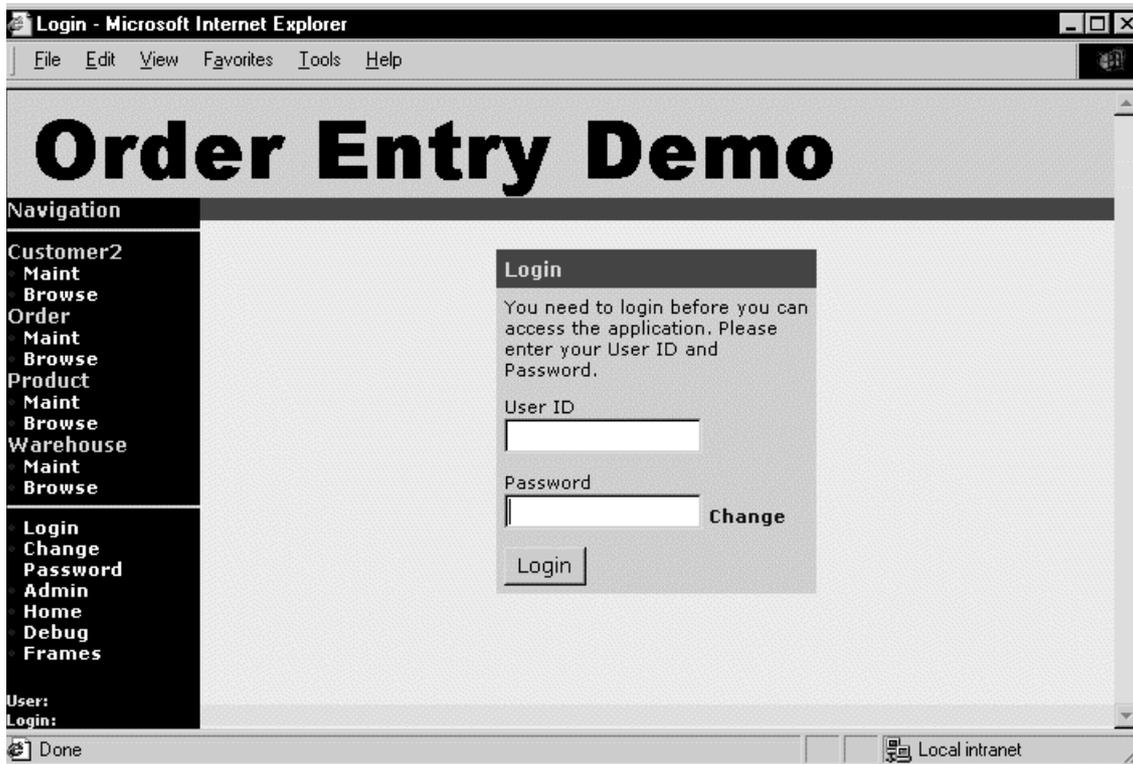
Navigation Bar

The navigation bar, another framework component, is included on every page in your application to contain links. When your web application is opened, the navigation bar is built dynamically, showing standard options: Login, Change Password, Admin, Home, Debug, and Frames. After the user logs in to the application, the navigation bar is populated with the business objects that the application uses and the functions, such as maintenance and browse, that are supported for each business object. The presence of objects and functions in the navigation bar is controlled by the Spectrum security settings that grant or deny the user access to objects and methods.

To customize the navigation bar, work with the page handler `NavBar.cls` and `NavBar.htm` that Construct Spectrum places in the web application directory.

Login Page

- To access business objects in the Demo application:
 - 1 Click **Login** in the navigation bar.
The Login page is displayed:



Login Page

- 2 Supply any user ID.

3 Click **OK**.

The Demo application is configured so that any user ID and a blank password give users access to the application.

If the call to the mainframe server is successful, the Home page is displayed and the navigation bar displays the actions you can perform for the business objects, for example, the Customer object has Maint and Browse links below it.

If the log in was not successful, the error is explained in the message area of the page. If the problem is caused by an incorrect dispatch service, see **Administration Page**, page 44.

Once you have logged in, **Login** changes to **Logout** on the home page and navigation bar. As well, the links to the application pages appear in the navigation bar. When you click **Logout**, the Login page is displayed and the options in the navigation bar return to pre-authenticated status.

Note: You will be logged out automatically if you do not use the application within the session timeout setting for the application. You can modify the application's session timeout setting using 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 so that the next time, the ID is displayed in the Login page.

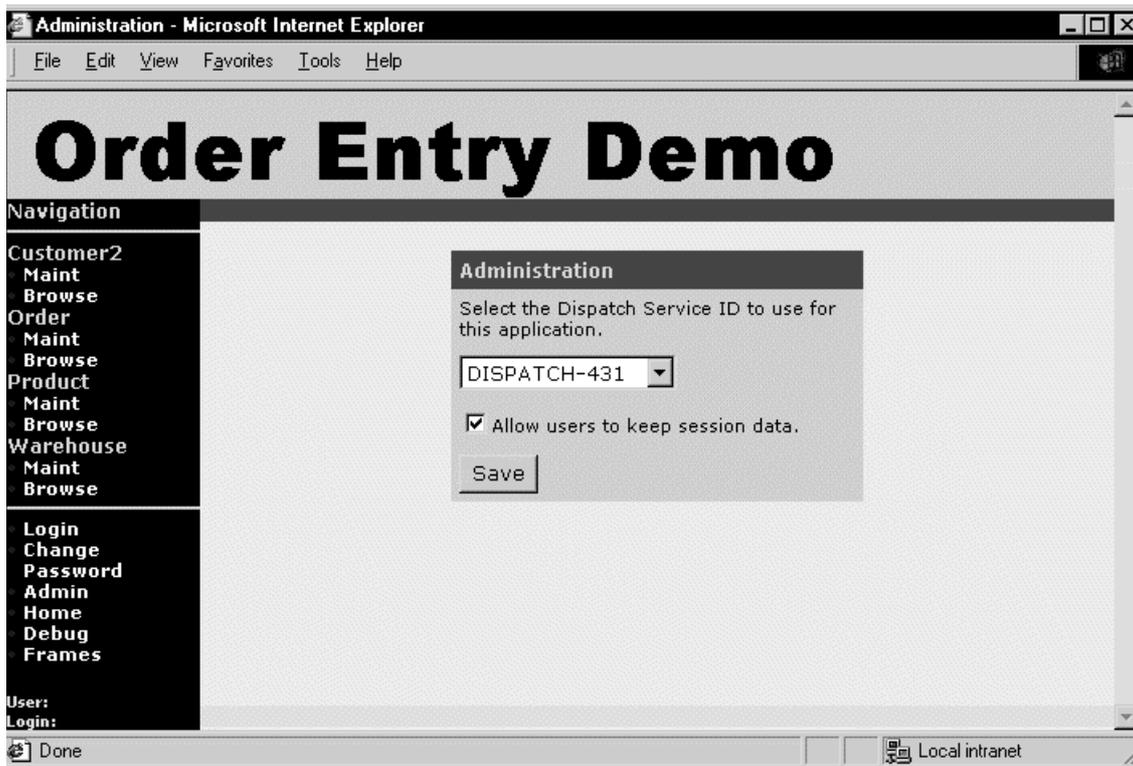
Spectrum web applications support securing the Login page using a secure web protocol (HTTPS) while allowing the rest of the application to use a normal unsecured protocol (HTTP). This ensures that a user's login ID and password are encrypted when sent over the network when the user logs on. By default, new web applications do not use this mode, but, by making a modification to a framework component, you can enable the secure login functionality. For more information, see **Securing Your Application**, page 179.

To customize the Login page, work with the page handler Login.cls and Login.htm. When creating the page handlers for your application, you can specify whether or not the user must log in to your application by selecting an option in the Page Handler wizard. For more information, see **Generating and Customizing Page Handlers**, page 85.

Administration Page

The Administration page is a framework component supplied with Construct Spectrum web projects. Using this page, you can view and change the Spectrum dispatch service used by the web server to access the mainframe server.

- To access the Administration page, click **Admin** in the navigation bar.



Administration Page

The drop-down list box allows you to select a dispatch service to be used by the web server to access the mainframe.

The **Allow user to keep session data** option determines what happens to users' session data if you change the dispatch service during operation. For example, if the dispatch service is modified to access a different library or database, this option controls what happens to data cached in users' sessions.

Select **Allow user to keep session data** to allow users to keep working with current session data. 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 displaying informing the user.

If the option is not selected, the users' session data is discarded, perhaps causing them to lose work.

When creating applications, you may wish 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 in the navigation bar, see **Securing Your Application**, page 179.

To customize the Administration page, work with the page handler `Admin.cls` and `Admin.HTML` that Construct Spectrum places in the web application directory.

Debug Page

The Debug page displays information about global performance and the user's current session. This page is a convenient tool when you are testing applications because it displays information about the objects accessed by the application and the cached errors.

- To access the Debug page, click **Debug** in the navigation bar.

Application Debug Data - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Order Entry Demo

Navigation Application Debug Data

Application Key	Value
Dispatcher.id	DISPATCH-431

Session Key	Value
~browser.level	1
~current.pageid	AppDebug
~messages	
~userid	User X
~password	
~session.time	9/27/99 1:51:42 PM
~refresh.frames	1

User:
User X
Login:

Done Local intranet

Debug Page

Before your application is distributed to users' machines, you may wish to remove the Debug page. However, it can be a useful tool for support personnel in diagnosing users' problems.

When you have just logged on to the application, the Debug page shows the following information:

Key	Value
Application Key	Information that applies to the application, rather than an individual user's session.
User.count	The number of users currently logged on to the application.
Dispatcher.id	The name of the dispatch service currently used to communicate with the mainframe.
Keep.sessiondata	The current value of the Allow users to keep session data option on the Administration page.
Session Key	Information that is specific to the user's current session.
~browser.level	The web browser in use. A value of 1 indicates Internet Explorer. A value of 2 indicates Navigator.
~current.pageid	The name of the page handler for the Debug page.
~messages	Messages that are queued for display.
~security.profile	The user's access privileges for objects and methods in the current application. For example, <DEMO.CUSTOMER.GET> means that the user is allowed to perform the Get action to display a customer record. The syntax for these entries is domain.object.method. For more information, see Setting up Your Application Environment on the Mainframe , page 45, <i>Construct Spectrum Programmer's Guide</i> .
~userid	The user ID with which the user logged on to the application.
~password	The user's password. This text box is blank in the Demo application because it is configured not to use passwords.

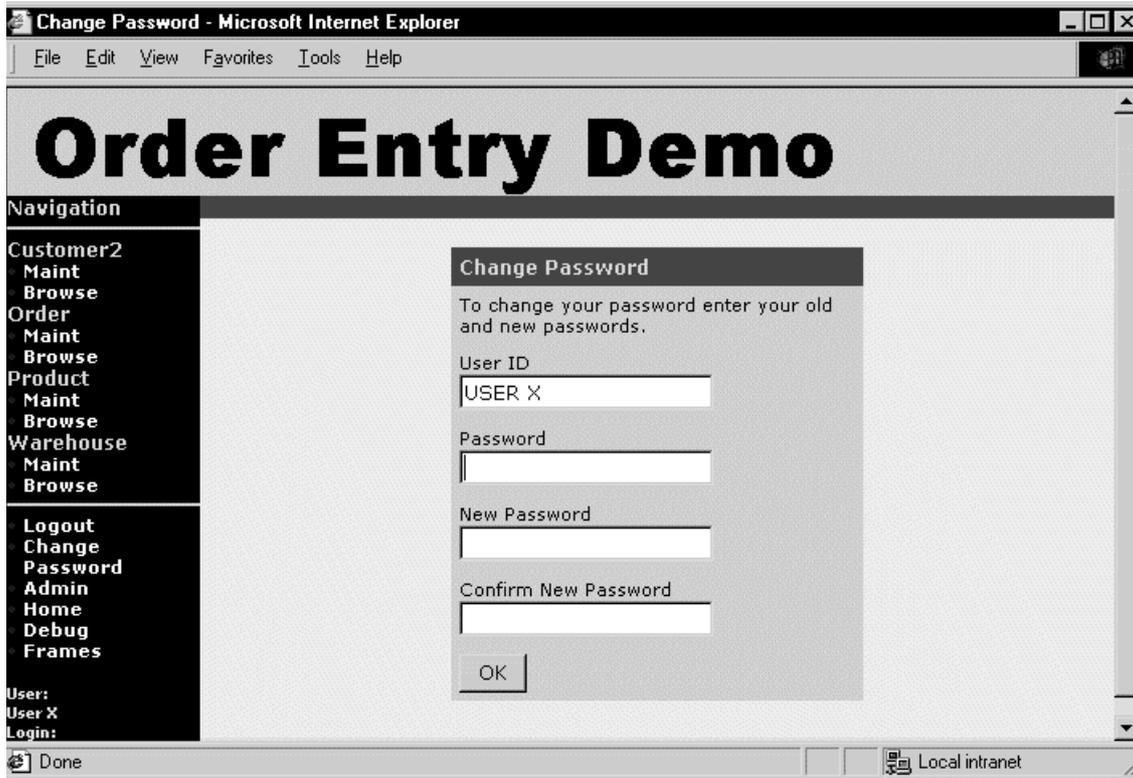
Key	Value (continued)
~session.time	The date and time at which the user logged in to the application.
~refresh.frames	A value of 1 indicates that frames need to be refreshed if Frames mode is on. A value of 0 indicates that frames are up to date.

If you are working with a maintenance or browse page that is accessing mainframe data, the Debug page displays more information. See **Validating Your Data**, page 171 for an example of the Debug page's ability to display cached errors.

To customize the Debug page, work with the page handler AppDebug.cls and AppDebug.htm that Construct Spectrum places in the web application directory.

Change Password Page

The Change Password page presents a form with which the user can change passwords for logging on to the web application:



Change Password Page

To customize the Change Password page, work with the page handler `ChangePassword.cls` and `ChangePassword.htm` that Construct Spectrum places in the web application directory.

Frames

The navigation bar contains an option that allows users to toggle between Frames and Nonframes mode. When Frames mode is on, the page is divided into independent sections that can scroll separately. This means, for example, that the header of the page and the navigation bar remain stationary if the user scrolls down in the main area of the page.

Nonframes mode is enabled by default. The following shows the Customer browse page when Frames mode is enabled:

The screenshot shows a web browser window titled "Customer2 Browse - Microsoft Internet Explorer". The main content area features a large heading "Order Entry Demo". Below the heading is a navigation sidebar on the left with a tree view containing items like "Customer2", "Maint", "Browse", "Order", "Product", "Warehouse", "Logout", "Change Password", "Admin", "Home", "Debug", and "Non-frames". To the right of the sidebar is a search area with a "Sort Key" dropdown set to "CustomerNumber", a "CustomerNumberKey" input field, and a "Range" dropdown set to "*". Below these are "Go" and "More" buttons, and a checkbox for "Display all fetched rows". The main content area displays a table with the following data:

#	Customer Number	Business Name	Phone Number	Credit Rating	Credit Limit
1	1	SAGA SOFTWARE INC (CANADA)	(519) 622-0889 C		\$2,000.00
2	2	JOURNEYMEN FABRICATING	(519) 234-6422 z		\$500,000.00
3	511	SOFTWARE IS US	(519) 624-5623 AAA		\$10,000.00
4	10001	Journeyman Fabricating	(519) 234-6422 z		\$1,000.00
5	10002	Les Rivers Custom Fabricating	(519) 623-6850 A		\$1,000.00
6	10003	MONJEN STELL INC.	(519) 624-5623 A		\$1,000.00
7	10004	STEELFALCO INC.	(519) 623-6850 A		\$1,000.00
8	10005	CAMBRIDGE T.V. STEREO	(519) 622-2425 A		\$2,000.00

The browser status bar at the bottom shows "Done" and "Local intranet".

Frames Mode

To customize the Frames option, work with the page handler `Frameset.cls` and `Frameset.htm` that Construct Spectrum places in the web application directory.

Features of a Maintenance Page

The Order Maintenance page contains examples of the functionality that Construct Spectrum provides as framework components, that are added to the application when its components are generated, and that can be added with custom code.

For an illustration of how framework HTML templates work together to present a page, see **Framework HTML Templates and Page Handlers**, page 77.

- To access this page, click the **Maint** link under **Order** in the navigation bar. The Order Maintenance page appears. Click **Next** to display a record:

The screenshot shows a web browser window titled "http://dbg/WebOrderEntry/CstOrderEntryDemo/WebApp.ASP?Page=Order.Maint&Src=Order%2EBrows...". The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. The main content area is titled "Order Entry Demo" and is divided into several sections:

- Navigation:** A vertical sidebar on the left contains links for Customer2, Order, Product, Warehouse, Logout, Change Password, Admin, Home, Debug, and Frames.
- Order Maint:** A form with input fields and buttons for "Find":
 - Order Number: 991
 - Order Amount: \$1,500.00
 - Order Date: 12/17/98
 - Customer Number: 33333 (TORONTO BLUE JAYS)
 - Warehouse ID: 544 (SELNIK STORAGE)
 - Invoice Number: 111111
- DeliveryInstructions:** A section with a scrollable text area.
- Order Lines:** A section with buttons for Read, Next, Clear, Add, Update, and Delete.
- Next Order displayed:** A section for displaying the next order.

At the bottom left, the user information is displayed: User: USER X, Login: Sep 27 03:42 pm. The bottom right shows a "Local intranet" icon.

Order Maintenance Page

The following sections explain some of the features of the Order Maintenance page.

Header

Order Entry Demo

Header

The header is a framework component supplied with Construct Spectrum web projects. To customize it, work with the page handler Header.cls and Header.htm that Construct Spectrum places in the web application directory.

Sections



Index	Product ID	Quantity	Unit Cost	Total Cost
1	187361	10	\$150.00	\$1,500.00

Order Lines Section

Whenever a periodic group, multi-value field, or related file is used in a Predict view, components of the group are generated in a table called a “section.” The name of the group becomes the section heading, as in Delivery Instructions, Order Lines, and Order Distributions in the Order Maintenance page.

When creating your own applications, you can modify the captions for sections and fields using the HTML Template wizard. See **Customizing HTML Before Generation**, page 123.

Collapsible Sections

For users of Internet Explorer, sections are “collapsible” so that the user can toggle between visible and invisible using the plus (+) and minus (-) icons. This option is useful for collapsing page content that the user currently doesn’t need, reducing 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. The Order Header section in the Order maintenance page displays the default single edit view, in which one record at a time is displayed and the user can modify all fields.

In the case of a related file, such as Order Lines or Line Distributions in the Order Maintenance page, drop-down lists and action buttons are also supplied in the single edit view. The user can select different records in the file and update them. The related file’s records are updated when the user clicks **Update**.

When you define a section with single edit view, you can offer users the additional option of toggling between single edit view and report view by clicking **View**. For example:



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

Report View

Multiple edit view presents all records in a table that the user can edit. In the case of an Alpha MU field, you can present the section as a text area that users can edit.

When creating your own applications, you can set the view options in the HTML Template wizard. For more information, see **Customizing HTML Before Generation**, page 123.

Find Buttons

The **Find** buttons in the Demo application are the result of customizations made in the HTML Template wizard before generating the template. Clicking **Find** invokes a browse page to select a value and return to the original maintenance page. Try clicking **Find** next to the **Customer** text box, for example, to view the Customer Browse page:

#	Customer Number	Business Name	Phone Number	Credit Rating	Credit Limit
1	1	SAGA SOFTWARE INC (CANADA)	(519) 622-0889 C		\$2,000.00
2	2	JOURNEYMEN FABRICATING	(519) 234-6422 z		\$500,000.00
3	511	SOFTWARE IS US	(519) 624-5623 AAA		\$10,000.00
4	10001	Journeyman Fabricating	(519) 234-6422 z		\$1,000.00
5	10002	Les Rivers Custom Fabricating	(519) 623-6850 A		\$1,000.00
6	10003	MONJEN STELL INC.	(519) 624-5623 A		\$1,000.00
7	10004	STEELFALCO INC.	(519) 623-6850 A		\$1,000.00
8	10005	CAMBRIDGE T.V STEREO	(519) 623-2435 A		\$2,000.00
9	10006	PAUL'S VINEYARDS	(519) 233-6230 A		\$1,000.00
10	10007	STANDARD TOOL AND DIE	(519) 623-1665 A		\$1,000.00
11	10008	TOYO TIRE CANADA, INC	(519) 623-2340 A		\$1,000.00
12	10009	AUTOWORKS	(519) 623-6850 A		\$1,000.00
13	10010	TIEN SUN	(519) 623-6850 A		\$1,000.00

Customer Browse

Select a customer by clicking a red arrow. The Order Maintenance page appears with the select customer number in the Customer text box.

For information about linking browse pages to maintenance pages, see **Customizing HTML Before Generation**, page 123.

Calendar Pop-Up

Information of type Date is automatically accompanied by a calendar pop-up when it is presented in a maintenance page. The user clicks the icon to display the pop-up and select a date. The user can select a month and year using the pull-down list boxes and then select a date by clicking the day in the calendar.

The calendar pop-up is a framework component supplied with Construct Spectrum web projects. To customize it, work with Calendar.htm that Construct Spectrum places in the web application directory.

Action Bar

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



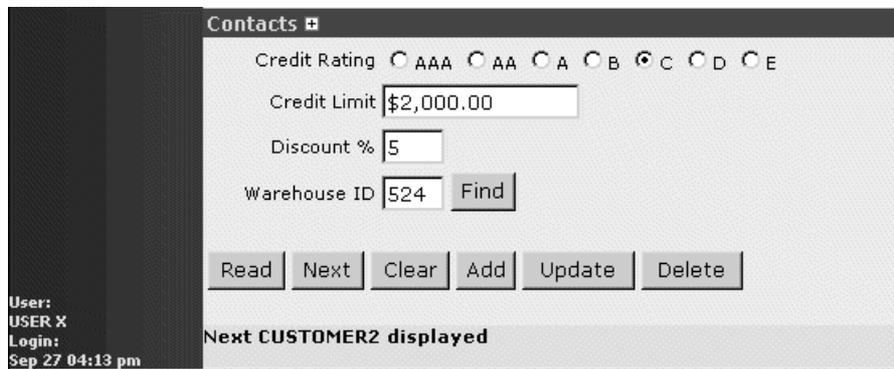
Action Bar

The contents of the action bar can vary depending on the user's access privileges. For example, a user who is allowed to view orders but not update them will not see the **Update**, **Add**, or **Delete** buttons on the maintenance page. For more information, see **Securing Your Application**, page 179.

The action bar is a framework component supplied with Construct Spectrum web projects. To customize it, work with the page handler ActionBar.cls and ActionBar.htm that Construct Spectrum places in the web application directory.

Message Area

Messages are displayed at the bottom of each page, for example:



The screenshot displays a web application interface. On the left, a dark sidebar contains the text: "User: USER X", "Login:", and "Sep 27 04:13 pm". The main content area is titled "Contacts" and features a form with the following fields and controls:

- Credit Rating: Radio buttons for AAA, AA, A, B, C (selected), D, and E.
- Credit Limit: A text input field containing "\$2,000.00".
- Discount %: A text input field containing "5".
- Warehouse ID: A text input field containing "524" and a "Find" button.
- Navigation buttons: "Read", "Next", "Clear", "Add", "Update", and "Delete".

At the bottom of the form, a message area displays the text: "Next CUSTOMER2 displayed".

Message Area

The message area is a framework component supplied with Construct Spectrum web projects. To customize it, work with the page handler `Messages.cls` and `Messages.htm` that Construct Spectrum places in the web application directory.

Features of a Browse Page

The Customer Browse page contains many of the framework HTML templates described in **Features of a Maintenance Page**, page 52: Header, sections, Footer, and Message area. In addition, it contains functionality that is specific to browse pages.

- To access the Customer Browse page:
 - 1 Click the **Browse** link under **Customer** in the navigation bar. The Customer Browse page appears.
 - 2 Click **Go**. The first 20 rows (records) are fetched. Because there are more records in the file, the **More** button appears, as well as the **Display all fetched rows** check box.

Order Entry Demo

Navigation **Customer2 Browse**

Sort Key: CustomerNumber

CustomerNumberKey:

Range: *

Go More Display all fetched rows

#	Customer Number	Business Name	Phone Number	Credit Rating	Credit Limit
1	1	SAGA SOFTWARE INC (CANADA)	(519) 622-0889 C		\$2,000.00
2	2	JOURNEYMEN FABRICATING	(519) 234-6422 z		\$500,000.00
3	511	SOFTWARE IS US	(519) 624-5623 AAA		\$10,000.00
4	10001	Journeymen Fabricating	(519) 234-6422 z		\$1,000.00
5	10002	Les Rivers Custom Fabricating	(519) 623-6850 A		\$1,000.00
6	10003	MONJEN STELL INC.	(519) 624-5623 A		\$1,000.00
7	10004	STEELFALCO INC.	(519) 623-6850 A		\$1,000.00
8	10005	CAMBRIDGE T.V STEREO	(519) 623-2435 A		\$2,000.00
9	10006	PAUL'S VINEYARDS	(519) 233-6230 A		\$1,000.00
10	10007	STANDARD TOOL AND DIE	(519) 623-1665 A		\$1,000.00
11	10008	TOYO TIRE CANADA, INC	(519) 623-2340 A		\$1,000.00
12	10009	AUTOWORKS	(519) 623-6850 A		\$1,000.00
13	10010	TIEN SUN	(519) 623-6850 A		\$1,000.00
14	10011	TIEN SUN	(519) 623-6850 A		\$1,000.00
15	11111	QUAKER OATS	(212) 312-3113 AAA		\$200,000.00

Local intranet

Customer Browse Page

The following sections explain the features specific to browse pages.

KeySelector Template

The KeySelector modules (KeySelector.cls and KeySelector.htm) are framework components Construct Spectrum places in the application web directory. The KeySelector page handler queries the browse ABO to ascertain the browse key components. The KeySelector template is responsible for the following features.

Multiple Sort Keys

Construct Spectrum browse pages support multiple sort keys, presenting them in a drop-down list box from which users can select.

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

Range Options

Construct Spectrum browse pages are generated with range option support for selecting records. The options are presented in the **Range** drop-down list. After you have selected a range option, click **Go** to update the display.

Go and More Buttons

Browse pages are generated with a **Go** button that retrieves up to 20 rows from the database to display in the browse page. If there are more records in the file, the **More** button and **Display all fetched rows** check box are displayed. The **More** button retrieves the next 20 rows. The user has the option of displaying all records that have been fetched from the file.

Browse Rows

Records are displayed in a table with column headers showing field names.

Fields can be linked to maintenance pages if they exist in the application. For example, because the Demo application includes a Customer Maintenance page, users can click a Customer ID on the Customer Browse page to open the maintenance page.

The links are defined using the HTML Template wizard. For more information, see **Customizing HTML Before Generation**, page 123.

CREATING A WEB PROJECT

Before generating the components of your web application, you must have access to the ABO project and generated ABOs. You must also create a Construct Spectrum web project to contain the web components of the application you will generate later on. This chapter explains how to use the Spectrum Web Project wizard to create your web project.

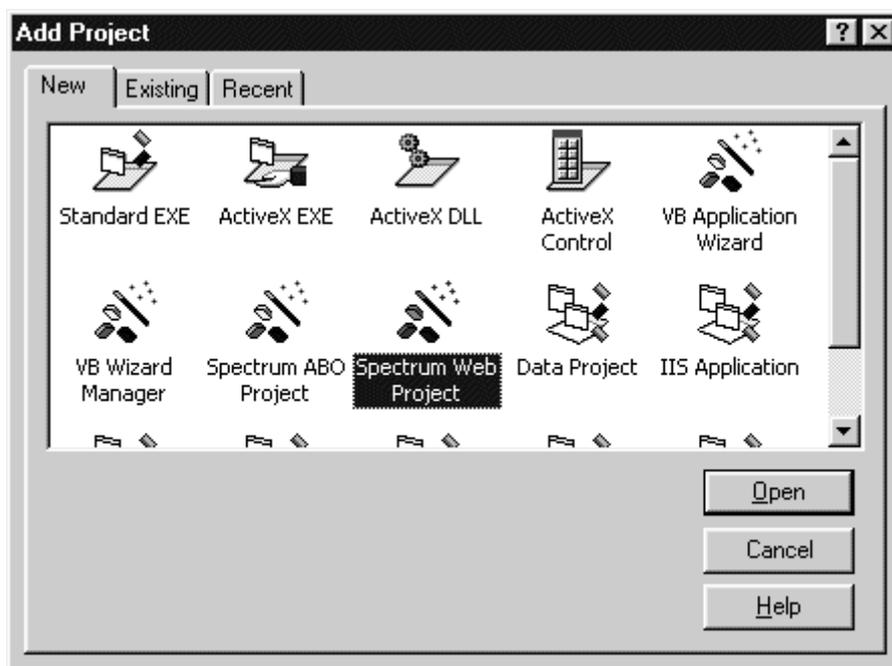
The following topic is covered:

- **Using the Spectrum Web Project Wizard**, page 64

Using the Spectrum Web Project Wizard

The web project contains the page handlers, HTML templates, and 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 From the **File** menu, select **Add Project**.
The **Add Project** dialog box appears. The **New** tab is displayed by default:



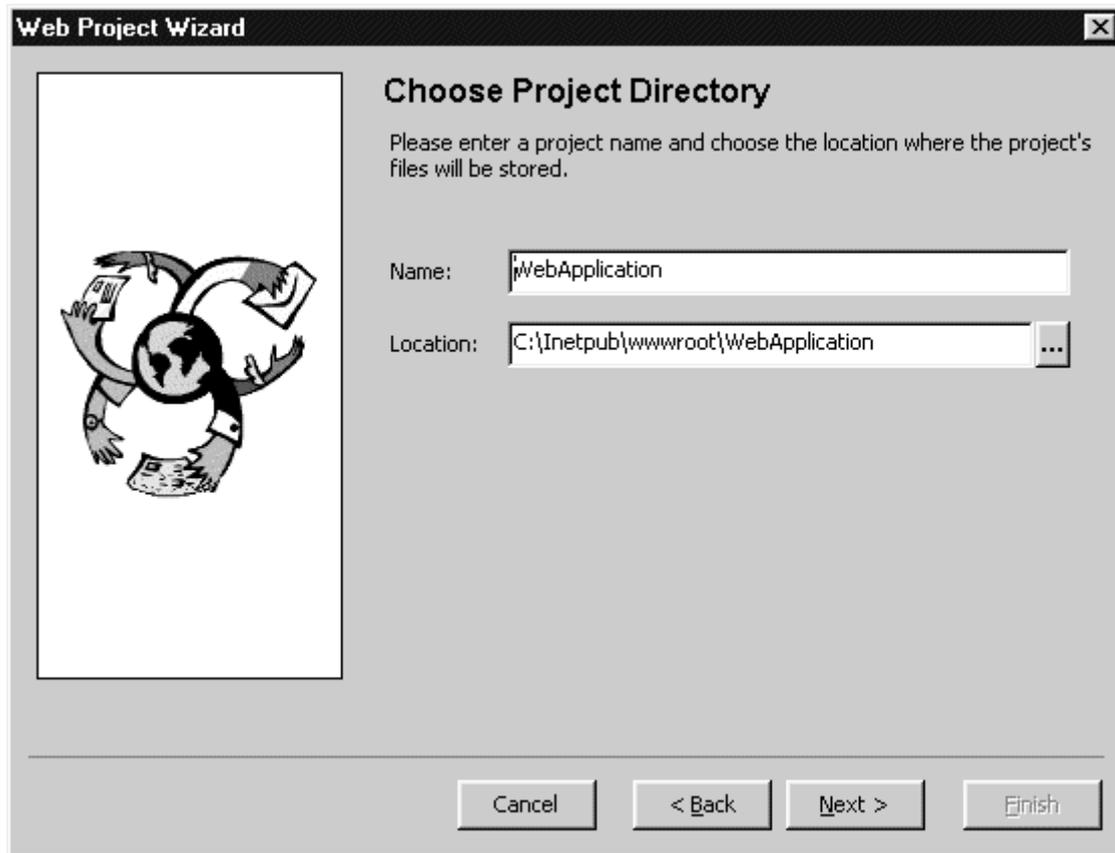
Add Project Dialog

- 3 Double-click **Spectrum Web Project**, or select it and click **Open**.
The Web Project wizard opens:



Web Project Wizard

- Click **Next**.
The **Choose Project Directory** step appears:

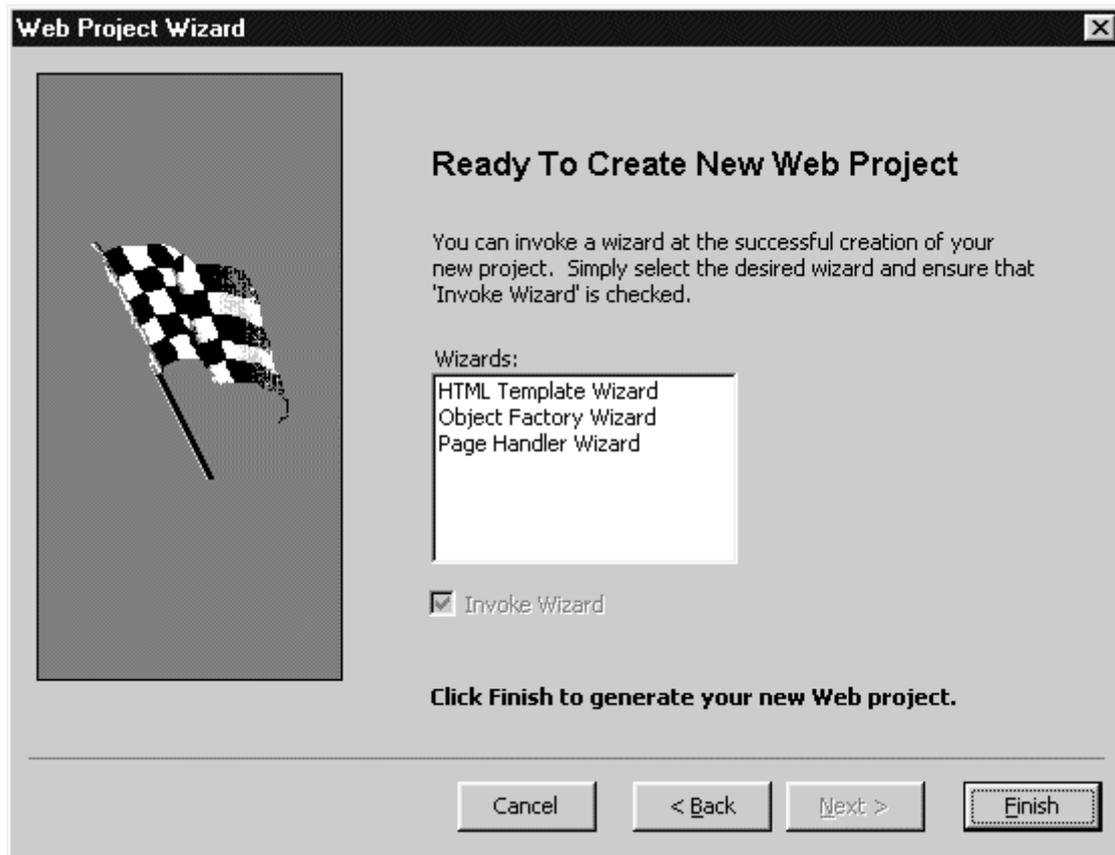


Web Project Wizard — Choose Project Directory

- Provide a name for the web project and specify a directory where the project files will be stored.

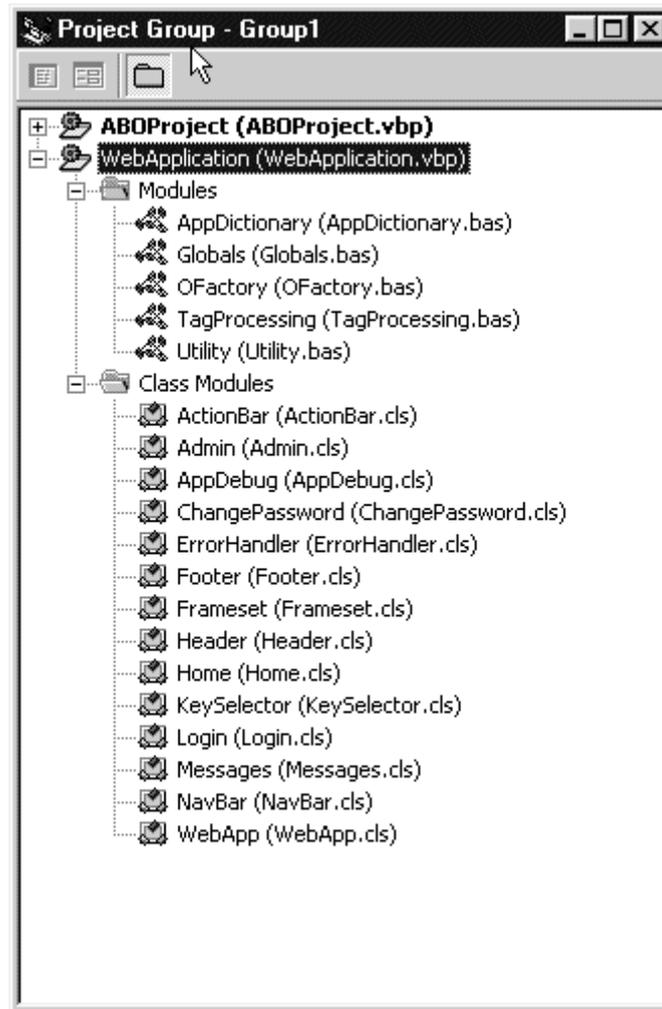
*Note: We recommend that you use C:\Inetpub\wwwroot as the project directory. If you do not use this directory to store the web components, you will need to add a virtual directory in IIS when you deploy the application. For more information, see **Deploying Your Web Application**, page 185.*

- 6 Click **Next**.
The **Ready to Create New Web Project** step appears:



Web Project Wizard — Ready to Create New Project

- 7 If you want to begin generating the web components for your application, select the wizard you want to invoke after you have generated the web project and select **Invoke Wizard**.
- 8 Click **Finish**.
The web project is generated and added to the project group:



Visual Basic Project Explorer

For information about the framework components, see **Framework Components Supplied with Construct Spectrum**, page 71.

You can now create the following components using the appropriate wizards:

- Page handlers
- HTML templates
- Object factory entries

FRAMEWORK COMPONENTS SUPPLIED WITH CONSTRUCT SPECTRUM

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

The following topics are covered:

- **Introduction**, page 72
- **Active Server Components and the WebApp.cls**, page 73
- **BAS Files**, page 74
- **Cascading Style Sheets**, page 76
- **Framework HTML Templates and Page Handlers**, page 77
- **JavaScript Files**, page 82

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 files, graphics, code frames, JavaScript, and cascading style sheets.

Framework components make it possible for you to generate a functional web application using only the Spectrum wizards and an existing ABO. After you have generated the web components for a business object (page handler, HTML template, and object factory entries), you can customize both generated and framework components to suit the purpose and design of your web application. You can also perform global customizations to framework components that will affect how generated components look and perform.

The following sections list the framework components and briefly describe their functions.

Active Server Components and the WebApp.cls

An Active Server Page (ASP) script is used to activate the WebApp.cls module, which in turn activates a specific web page. The ASP passes information obtained from the web server, such as session and application objects, to a web application associated with the ASP script. When you update the object factory, these files are also updated.

The following files are supplied as framework components to support your application:

File	Description
Global.asa	This ASA detects session and application events. It is used to determine when a user starts a session and when that user's session ends. It can also detect when the web application starts for the first time and when it ends.
WebApp.asp	This ASP accesses the web application in Nonframes mode.
WebAppF.asp	This ASP accesses Frames when the application is in Frames mode.
WebAppFS.asp	This ASP requests the frame set. The frame set describes the layout of web pages in Frames mode.
WebApp.cls	This module is the public creatable component in the application, providing a starting point for requests to the application. It is called by an ASP when a web browser requests the page. It includes separate routines for Frames and Nonframes modes.

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

BAS Files

A number of BAS files are included in the web project to provide global functions:

File	Description
AppDictionary.bas	The dictionary file contains translations for terms used in HTML pages.
OFactory.bas	The object factory co-ordinates the creation of objects during execution of the web application. It also contains security routines to validate the user and evaluate security tags in HTML templates. You can update the object factory using the Object Factory wizard. For more information, see Updating and Customizing the Object Factory , page 163.
Globals.bas	The Globals file includes settings and routines used throughout the web application.
TagProcessing.bas	The tag parser contains generic tag processing used by the whole application. Also contains a routine you can customize to introduce new tags. For more information, see Construct Spectrum Replacement HTML Tags , page 145.
Utility.bas	This file contains utility and helper routines.

Customizing BAS Files

You can apply global changes to your application by modifying support files. You are most likely to customize the following files:

- AppDictionary.bas
- Globals.bas
- TagProcessing.bas

AppDictionary.bas

When you create a new project, this file is automatically populated with the names for the actions that can be performed in the application. Use this file to provide substitute values for field names. For example, the key selector page handler (KeySelector.cls) calls AppDictionary.bas to determine whether it contains pre-defined names for the template to use. Since the key selector relies on descriptor names that are used as search criteria, you can specify meaningful names for the descriptor fields associated with the files accessed by the application.

Globals.bas

If you wish to enable the Login page in your application, you must modify Globals.bas. For more information, see **Globals.bas**, page 182.

TagProcessing.bas

Use the tag processing module to write customizations that all your applications can draw upon. Writing these customizations in this module makes them easy to maintain.

For more information, see **Construct Spectrum Replacement HTML Tags**, page 145.

Cascading Style Sheets

Construct Spectrum web applications use supplied cascading style sheets to format pages consistently. For example, the framework CSS defines the background colors for pages, fonts used in text, button colors, etc.

The following cascading style sheets are supplied:

File	Description
StylesIE.css	This style sheet defines the look of elements in web pages when they are viewed with Internet Explorer.
StylesNav.css	This style sheet defines the look of elements in web pages when they are viewed with Navigator.

Cascading style sheets define a set of styles that override the web browser's standard methods for rendering HTML. This gives your pages a unique and consistent design. Style sheets allow you to define the attributes of any tag. For example, you can adjust the font, line spacing, justification, and border properties.

Style sheets are applied to a page by adding a <LINK> tag in the HTML document heading, between <HEAD> and </HEAD>. In Construct Spectrum web applications, you can find the style sheet links in Layout.htm, which determines the layout and appearance of all web pages in the application.

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

```
<LINK REL=stylesheet HREF="support\stylesNav.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.

Framework HTML Templates and Page Handlers

Several HTML templates are supplied as framework components to present standard page components, such as the navigation bar, that are common to web pages. Most HTML templates require page handlers to process their tags, for example, replacing replacement tags with live content or other HTML templates.

To view most of the following framework components, open the Demo web application in your browser or see **Features of the Demo Web Application**, page 37.

The following table lists and explains the files in alphabetical order:

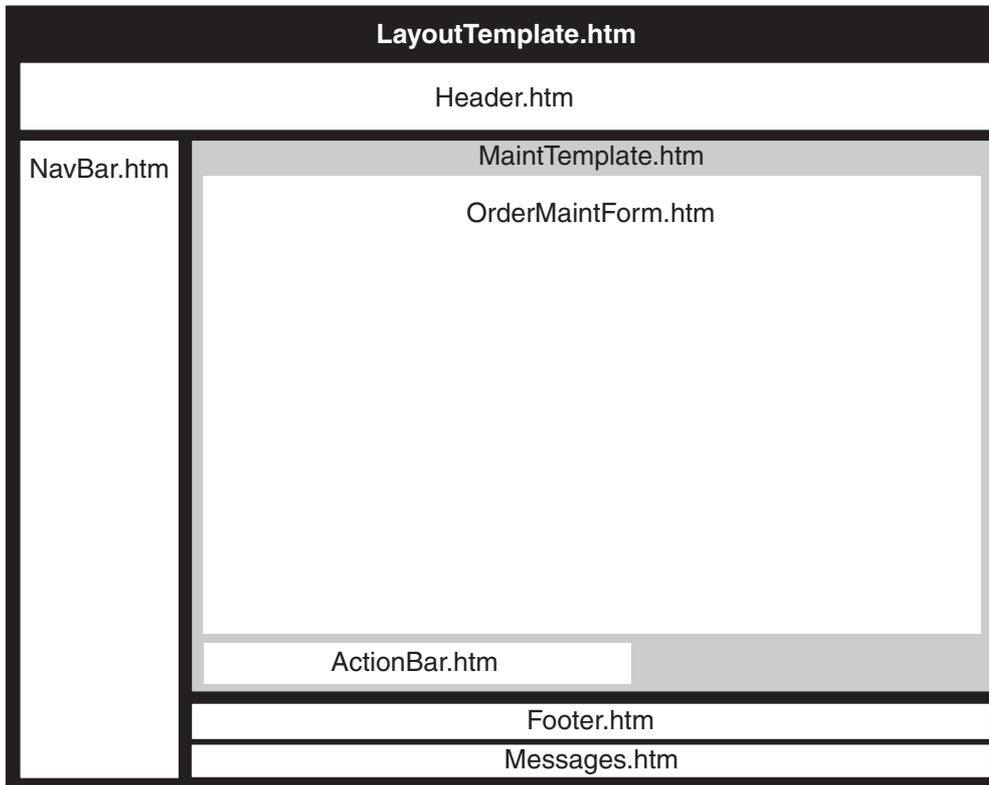
File	Description
ActionBar.cls ActionBar.htm	The action bar presents the methods available for the business object on a maintenance page. It displays buttons like Read , Next , Clear , Add , Update , and Delete .
Admin.cls Admin.htm	The Administration page is standard to Construct Spectrum web applications, allowing you to change the dispatch service used by the web server to access the mainframe. By default, the navigation bar contains a link to the Administration page.
AppDebug.cls AppDebug.htm	The Debug page is standard to Construct Spectrum web applications. It displays information about user and application settings that are cached on the web server. By default, the navigation bar contains a link to the Debug page.
BestViewed.htm	This page appears if the user's web browser does not meet the minimum requirements. Construct Spectrum web applications are best viewed using Internet Explorer 4.0 (or higher) or Netscape Navigator 4.0 (or higher).
BrowseTemplate.htm	This HTML template determines the layout of browse pages.

File	Description (continued)
Calendar.htm	Information of type Date is always accompanied by a calendar pop-up. The user can select a month and year using the drop-down lists and then select a date by clicking the day in the calendar.
ChangePassword.cls ChangePassword.htm	The Change Password page is standard to Construct Spectrum web applications, allowing users to change the passwords they use to log in to the application. By default, the navigation bar contains a link to the Change Password page.
ErrorHandler.cls ErrorHandler.htm	These files are responsible for displaying a standard response page whenever an invalid or unknown page is requested.
Footer.cls Footer.htm	The footer is a standard component of every web page. You can customize the HTML template to display, for example, links to other pages or your corporate logo.
Frameset.cls Frameset.htm	These files are responsible for the appearance and performance of web pages when Frames mode is on.
Header.cls Header.htm	The Header modules are responsible for displaying content that is standard for every page in your application, as well as custom content specified in the generated HTML template for the page.
Home.cls Home.htm	The Home page is standard to Construct Spectrum web applications, providing users with a starting point from which to access other pages.
KeySelector.cls KeySelector.htm	The key selector components are used by all browse pages to display generic key selection options. The page handler queries the browse at run-time and uses the information gathered to display the correct logical key selection and key components.
LayoutTemplate.htm	This HTML template determines the layout of web pages operating in Nonframes mode.

File	Description (continued)
Login.cls Login.htm	The Login page is standard to Construct Spectrum web applications, allowing users to access the web application's business objects. The page handler calls the object factory to validate the user ID and password.
Logout.htm	The Logout page is standard to Construct Spectrum web applications, appearing when the use logs out of the application.
MaintTemplate.htm	This HTML template determines the layout for maintenance pages.
Messages.cls Messages.htm	The messages area is a standard component of every web page. Located above the footer, it displays status and error messages.
NavBar.cls NavBar.htm	<p>The navigation bar is standard to Construct Spectrum web applications. Appearing on the right of each web page, it presents links to the other web pages in the application. By default, the navigation bar displays standard options such as Login, Logout, Change Password, Admin, Home, Debug and Frames. After the user logs in, the navigation bar is populated with the application's business objects and the functions that are available for them.</p> <p>The page handler builds the navigation bar dynamically by querying the object factory regarding a user's permissions to access business objects.</p>

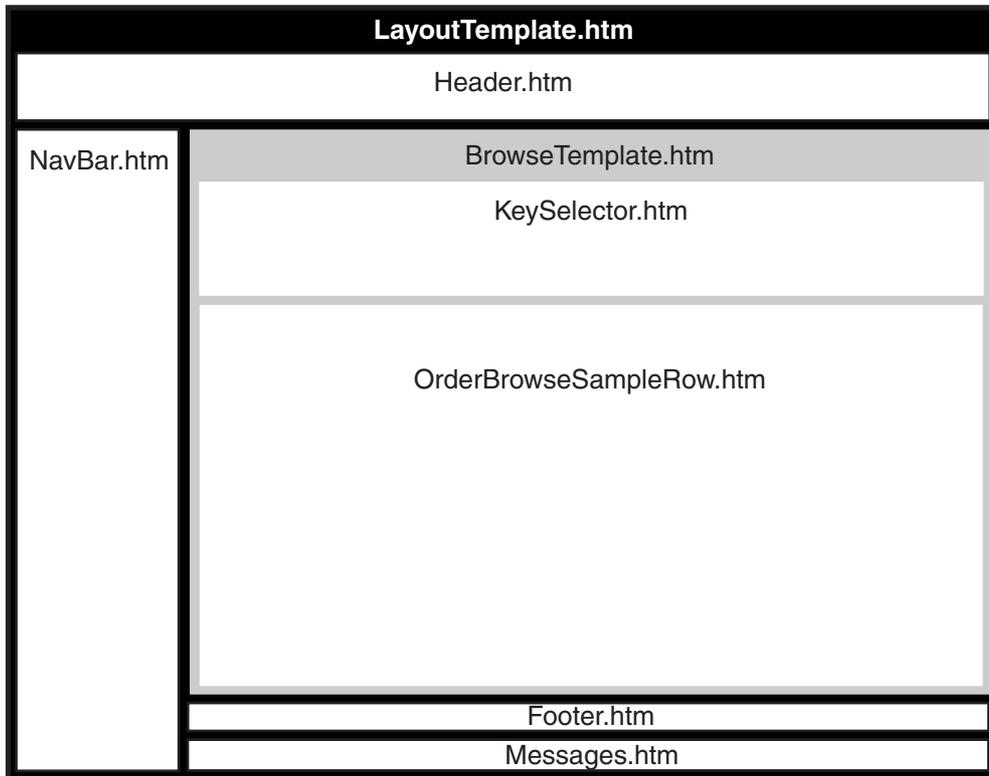
Examples of HTML Templates in Web Pages

The following illustration shows how generated and framework HTML templates work together to build a maintenance page:



HTML Templates in a Maintenance Page

The following illustration shows how generated and framework HTML templates work together to build a browse page:



HTML Templates in a Browse Page

Each of the HTML templates identified in the above examples are accompanied by page handlers. The only generated templates in the examples are those that present the actual content of the page: OrderMaintForm.htm and OrderBrowseSampleRow.htm.

JavaScript Files

Several JavaScript files are included in Construct Spectrum web projects to perform standard functionality.

File	Description
BrowseCommon.js	<p>This file contains scripts that provide common functions to browse pages. There are scripts to:</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.
Common.js	<p>This file contains scripts that provide common functions to all web pages. There are scripts to:</p> <ul style="list-style-type: none">• Get a cookie key's value.• Set a cookie key's value.• Enable or disable a form's elements.• 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.
IEBrowse.js	<p>This file contains scripts that provide common functions to browse pages when viewed with Internet Explorer. There are scripts to:</p> <ul style="list-style-type: none">• Copy text box values to the master form whenever the values change.• Save the key selector's visibility setting.• Set a section's visibility.• Show the currently selected logical key's fields.• Change an inner HTML value.

File	Description (continued)
IECommon.js	<p>This file contains scripts that provide common functions to all web pages when viewed with Internet Explorer. There are scripts to:</p> <ul style="list-style-type: none">• Bold graphics.• Fade graphics.• Hide or show a section.
IEMaint.js	<p>This file contains scripts that provide common functions to maintenance pages when viewed with Internet Explorer. There are scripts to:</p> <ul style="list-style-type: none">• Switch a view for a section by hiding or displaying two different sections.• Highlight form errors.
NavBrowse.js	<p>This file contains scripts that provide common functions to browse pages when viewed with Navigator. There are scripts to:</p> <ul style="list-style-type: none">• Change an inner HTML value.• Set a section's visibility.• Show the currently selected logical key's fields.
NavCommon.js	<p>This file contains scripts that provide common functions to all web pages when viewed with Navigator. There are scripts to:</p> <ul style="list-style-type: none">• Bold graphics.• Fade graphics.

GENERATING AND CUSTOMIZING PAGE HANDLERS

This chapter explains how to use the Page Handler wizard to generate page handlers for your web application and how to use the user exits supplied in page handlers.

The following topics are covered:

- **Using the Page Handler Wizard**, page 86
- **Customizing Page Handlers**, page 97

Using the Page Handler Wizard

A page handler is a Visual Basic class that is part of your web project. Page handlers respond to user requests, return HTML to the browser, and access ABOs to invoke business object properties and methods. Typically, you will generate a page handler for every ABO in your application.

Using the Page Handler wizard involves the following steps:

- Invoking the wizard
- Selecting an ABO
- Confirming the ABO's details
- Configuring the page handler
- Generating the page handler and saving it to your Visual Basic project, with the optional steps of viewing the generation report and comparing code.

The following sections explain the steps in detail.

Invoking 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:
- 1 Open the project group for your application in Visual Basic and select the web project in Project Explorer.
 - 2 From the **Spectrum** men, point to **Wizards** and select **Page Handler**. The Page Handler wizard appears:



Page Handler Wizard

For information about using the Spectrum Cache viewer  or Configuration editor , see **Features of the ABO and Web Wizards**, page 59, *Construct Spectrum Programmer's Guide*.

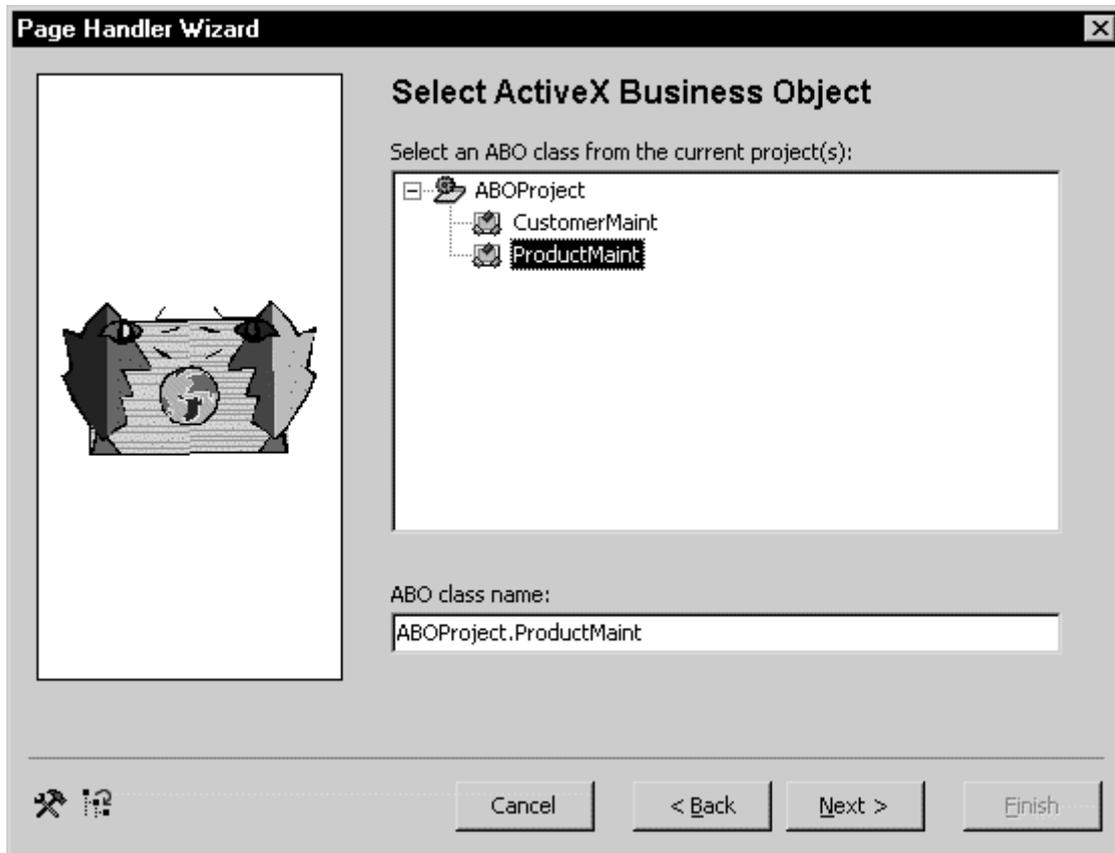
Selecting an ABO

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

➤ To select an ABO:

1 Click **Next**.

The **Select ActiveX Business Object** step appears:



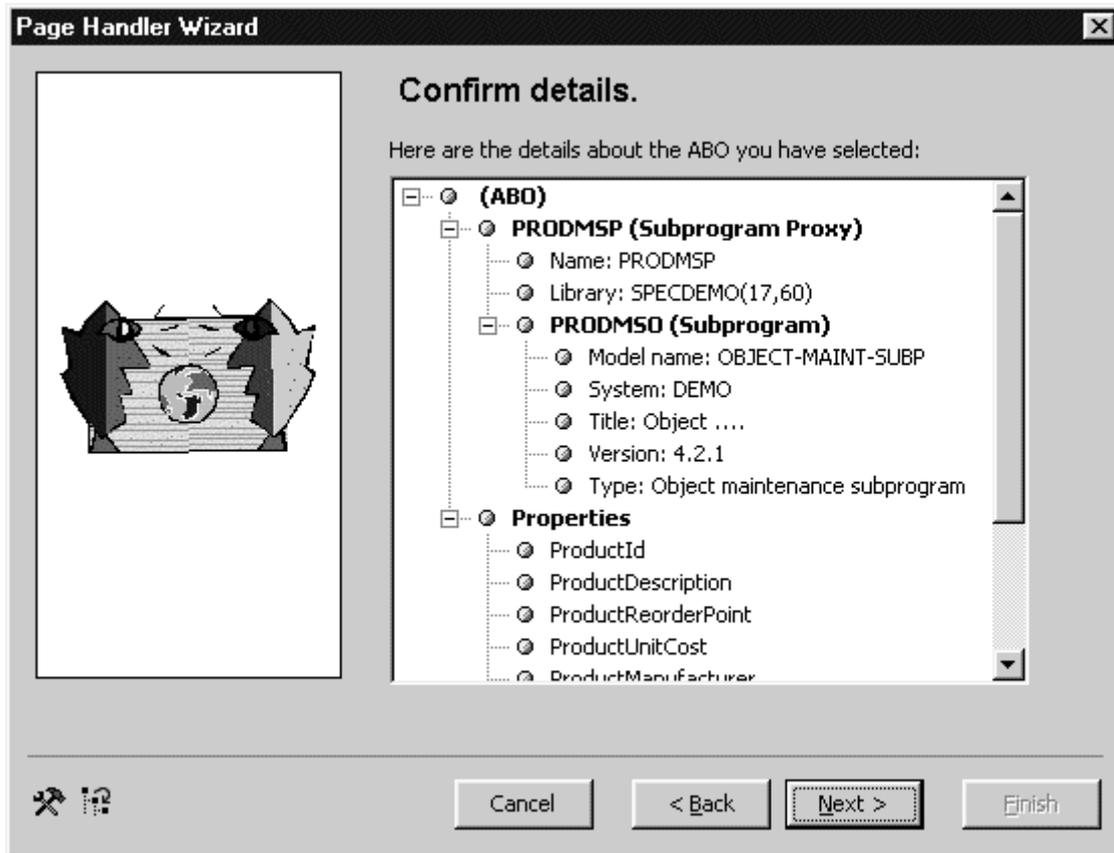
Page Handler Wizard — Select ActiveX Business Object

2 Select an ABO from the ABO project in the group currently open in Visual Basic.

3 Click **Next**.

Confirming ABO Details

When you select an ABO and click **Next**, the wizard presents a summary of the ABO:



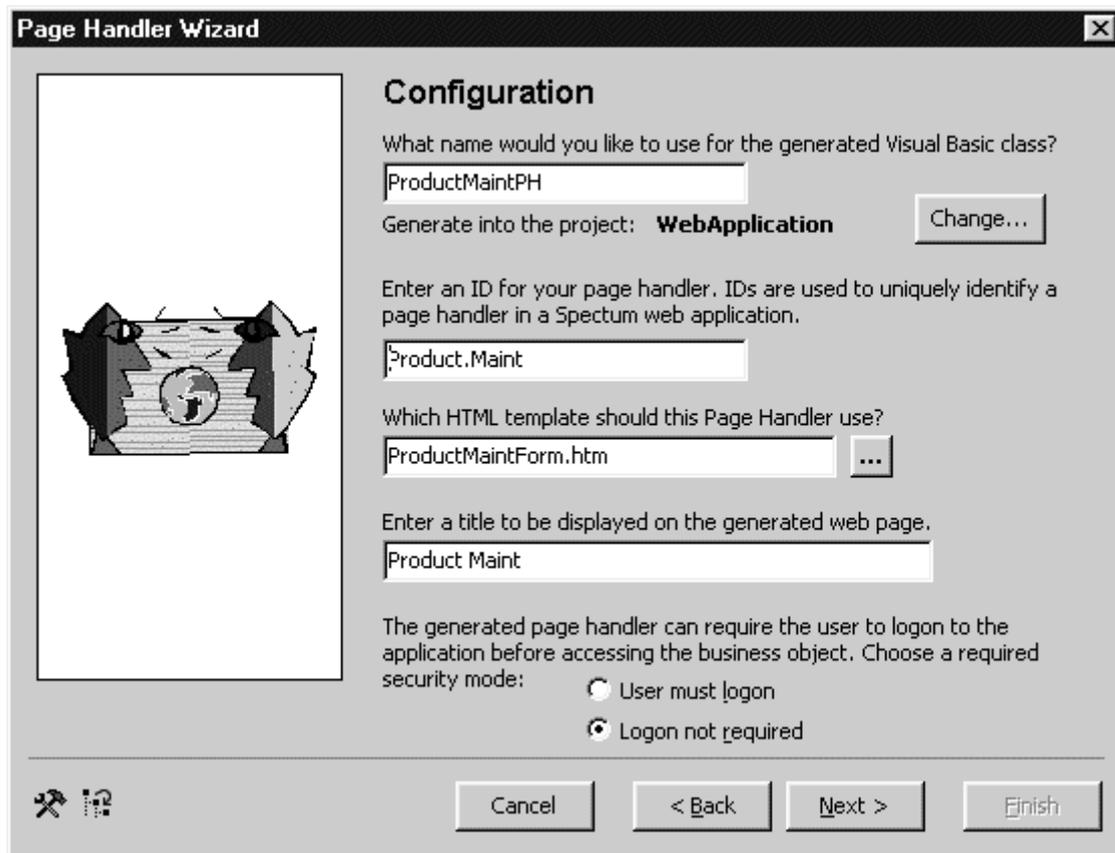
Page Handler Wizard — Confirm Details

The list displays the subprogram proxy and subprogram that the ABO encapsulates, as well as the properties associated with it.

- At this point, you can either:
- Click **Back** to select another ABO.
- Or
- Click **Next** to proceed to the next step.
The wizard performs the following steps:
 - Reads the ABO specification to confirm that the functionality it requires is supported. If the functionality isn't supported, a message appears prompting you to select a different ABO class.
 - Reads the field names and properties exposed by the ABO.

Configuring the Page Handler

When you select an ABO and click **Next** to confirm your selection, the wizard presents configuration options:



The screenshot shows the "Page Handler Wizard" dialog box in a "Configuration" step. On the left is a preview area with a broken document icon. The right side contains several configuration fields and options:

- Configuration**
- What name would you like to use for the generated Visual Basic class?
Text box: ProductMaintPH
- Generate into the project: **WebApplication** (with a "Change..." button)
- Enter an ID for your page handler. IDs are used to uniquely identify a page handler in a Spectrum web application.
Text box: Product.Maint
- Which HTML template should this Page Handler use?
Text box: ProductMaintForm.htm (with a "..." button)
- Enter a title to be displayed on the generated web page.
Text box: 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 are icons for help and a question mark, and buttons for "Cancel", "< Back", "Next >", and "Finish".

Page Handler Wizard — Configuration

➤ To configure the page handler:

- 1 Check the default name for the generated Visual Basic class and change it, if necessary.

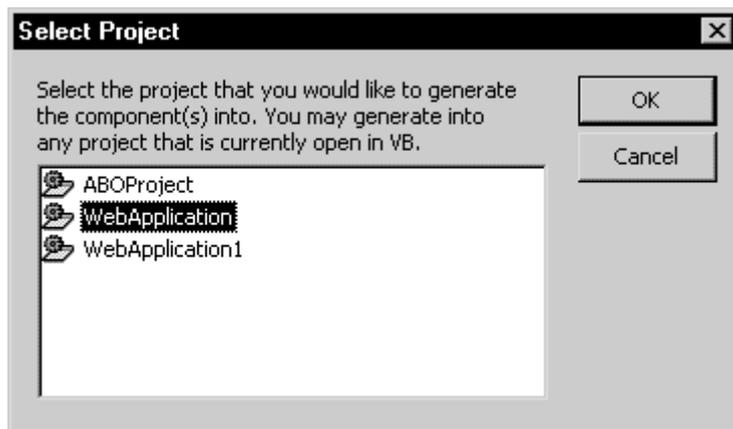
You can change the file name under which the page handler will be saved by typing the new name in the **File name** text box. The file will be saved in the same directory as your web application.

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.

Tip: You can change how default names are derived for page handlers using the Configuration editor. For more information, **The Configuration Editor**, page 60, *Construct Spectrum Programmer’s Guide*.

- 2 Check that the correct web project is selected and change it, if necessary.

By default, the page handler is saved to the web project that you selected in the Project Explorer before invoking the wizard. However, if you would like to save the file to another project, click **Change** to locate and select another project. The **Select Project** window appears:



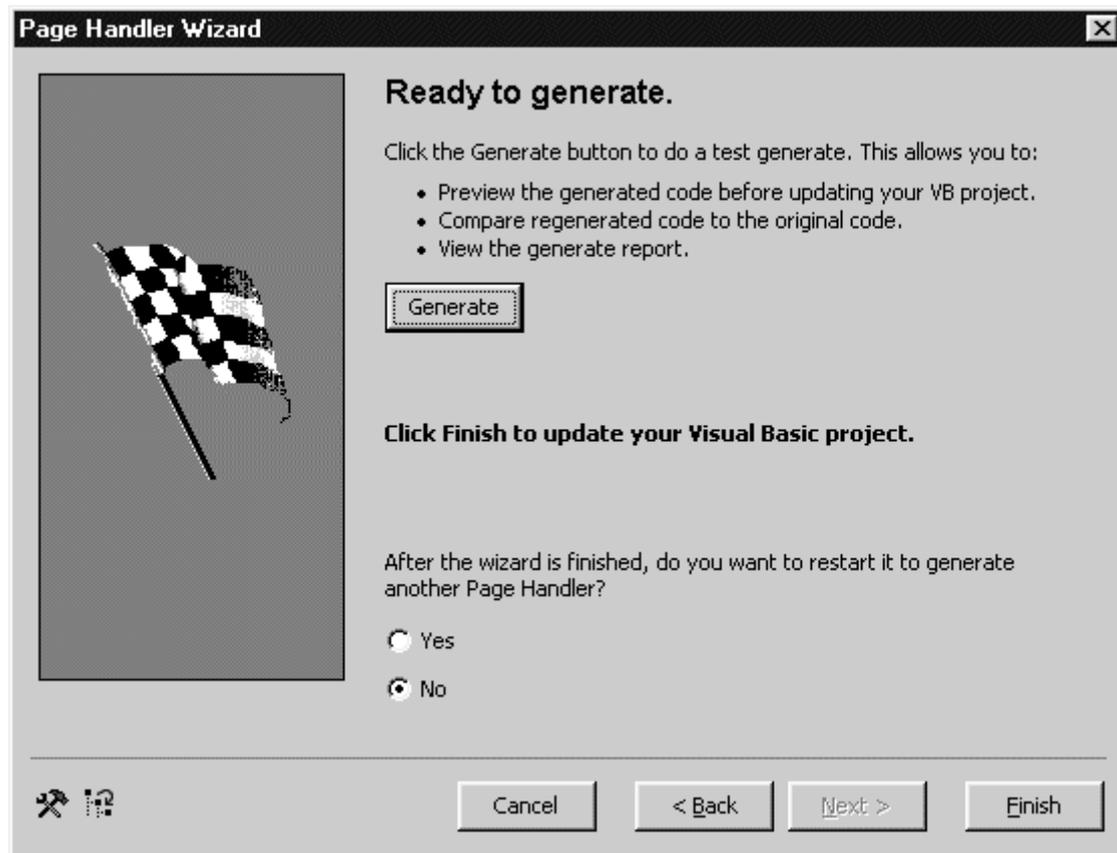
Select Project

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

- 3 Check the default ID for the page handler and change it, if necessary. This ID will be used, for example, to provides links between pages.
- 4 Check the default HTML template and change it, if necessary.
Click **...** to select a different template from a list of other templates you may want to use. For more information about HTML templates, see **Framework Components Supplied with Construct Spectrum**, page 71.
- 5 Specify the level of security you want to apply to the business object.
If you want users to be log on to the application before they can access this page, select **User must logon**.
- 6 If you want users to be able to access this page without being logged onto the application, select **Logon not required**.
For more information about using the Login functionality, see **Securing Your Application**, page 179.
- 7 Click **Next**.

Generating the Page Handler

When you click **Next** after configuring your page handler, the wizard presents generation options:



Page Handler Wizard — Ready to Generate

➤ At this point, you can perform three actions:

- 1 If you wish 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 you generated with code from an earlier generation of the module.

For information about using a code comparison utility with Construct Spectrum, see **Using Reports with a Code Comparison Tool**, page 82, *Construct Spectrum Programmer's Guide*.

For information about the generation report, see **Using Reports**, page 77, *Construct Spectrum Programmer's Guide*.

- 2 If you want to generate another page handler, select **Yes**.
- 3 When you wish to generate the code, update the Visual Basic project, and close the wizard, click **Finish**.

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

Customizing Page Handlers

You will probably wish to customize your application so that it performs tasks specific to your web application. This might mean modifying page handlers to customize security logic, or adding your own custom tags. For more information about customizing tags, see **Construct Spectrum Replacement HTML Tags**, page 145.

Customizing a page handler requires adding custom code to user exits or modifying generated code and then protecting it when regenerating. For more information about page handler user exits, see **User Exits in Maintenance Page Handlers**, page 97 and **User Exits in Browse Page Handlers**, page 101.

Protecting Generated Code

You can protect generated code during a regenerate using the `cst:PRESERVE` tag to surround the customized area. For more information about protecting generated code, see **Preserving Customizations to Generated Code**, page 71, *Construct Spectrum Programmer's Guide*.

Implied User Exits

You can add custom code to any function or subroutine using implied users exits. Implied user exits act as placeholders for hand-coded user exits during generation, ensuring that they are placed properly in the source code. For more information about implied user exits, see **Implied User Exits**, page 70, *Construct Spectrum Programmer's Guide*.

User Exits in Page Handlers

This section lists and describes how to use the user exits in maintenance and browse page handlers.

User Exits in Maintenance Page Handlers

The following user exits are supplied in maintenance page handlers.

ICSTPageHandler_Content.CustomContentIDs

You can use this exit to extend the standard template content and process new templates based on a content ID supplied to the program. The following example uses a template called OrderMaintForm2.htm whenever the ContentID Form2 has been specified.

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

ICSTPageHandler_BDTOverrides

Use this exit to change the BDT names for specific fields or to change how an ABO logical format is translated into a BDT name. The following example tells the application to use the Currency BDT for the field OrderAmount and the Alpha BDT for the OrderWarehouseID. It also specifies that the logical format Phone should use a Numeric BDT (the default translation is to use a BDT that is the same name as the logical format).

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

ParseTemplate.CustomTags

Using this exit, you can support new or customized tags added to your HTML template for this page handler. This example creates two new tags to use within any templates that this page handler supports. 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 **Construct Spectrum Replacement HTML Tags**, page 145.

PerformAction.OtherResets

This exit can be used to reset any custom values when the user retrieves a new record. In the example below, we reset a variable called m_Notes to an empty string.

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

PerformAction.CustomUpdateActions

This exit can be coded to add customized actions that require the framework to update the ABO from the information contained in the HTML template. The example shows how a custom action called RecalcDates will cause all of the data contained in the HTML page to update the ABO.

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

PerformAction.UpdateForeignKeys

This exit must be coded if your maintenance object supports foreign key lookups. The HTML template wizard can generate the necessary HTML elements to start the lookup, but this exit must be used to update the ABO with the fields returned

by the foreign key browse. The following example shows how the OrderWarehouseID field is updated when a foreign key browse was performed by the page handler, Warehouse.Browse.

```
'<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 your own data validations. This example shows how to add a validation to ensure that an order date is within some acceptable range (in this case greater than today). Note that it checks to see if a custom action called RecalcDates is being executed. This action updates this field, so we do not want any error checking to occur.

```
'<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")
                AddErrorMessage m_RequestData, m_ABOInterface
                Exit Sub
            End If
        End If
    End If
'</cst:EXIT>
```

PerformAction.CustomActions

Code this exit to handle custom maintenance actions. The following example shows how a custom action can be added to recalculate a date field. A button was added to 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

These two exits can be coded to allow the page handler to cache additional data in the session object. The following two examples show how the value of a module level variable can be stored and retrieved from the session object.

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

UpdateData.CustomUpdates

This exit can be used to update other data sources from the HTML template. This example shows how an item in the request object can be used 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>
```

User Exits in Browse Page Handlers

The following user exits are supplied with browse page handlers.

ICSTPageHandler_Process.CustomActions

This exit can be coded to include custom browse actions. The following example shows how a custom action called BROWSEALL can be added. This action retrieves all of the records in database before returning the HTML content to the user.

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

ICSTPageHandler_BDTOverrides

Use this exit to change the BDT names for specific fields or to change how a ABO logical format is translated into a BDT name. The following example tells the application to use the Currency BDT for the field OrderAmount and the Alpha BDT for the OrderWarehouseID. It also specifies that the logical format Phone should use a Numeric BDT (the default translation is to use a BDT that is the same name as the logical format).

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

RetrieveFromSession.CustomState and StoreToSession.CustomState

These two exits can be coded to allow the page handler to cache additional data in the session object. The following two examples show how the value of a module level variable can be stored and retrieved from the session object.

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

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

ParseTemplate.CustomTags

Using this exit you can support new or customized tags added to your HTML template for this page handler. This example creates two new tags to use within any templates that this page handler supports. The first tag, DUE_DATE, calculates a value based on a field in the 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,
m_ABOInterface.Field("OrderDate")), _
      "dd-mmm-yyyy")
    breplaced = True
  End If
Case "NOTES"
  tag.Contents = m_Notes
  breplaced = True
'</cst:EXIT>
```

For more information, see **Construct Spectrum Replacement HTML Tags**, page 145.

ICSTPageHandler_Content.CustomContentIDs

You can use this exit to extend the standard template content and process new templates based on a content ID supplied to the program. The following example uses a template called OrderMaintForm2.htm whenever the ContentID Form2 has been specified.

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

GENERATING AND CUSTOMIZING HTML TEMPLATES

HTML templates are the building blocks of your application's web pages. They present web pages or components of web pages that the page handler interprets and assembles at runtime. This chapter explains how to use the HTML Template wizard to create maintenance and browse pages.

The following topics are covered:

- **Introduction**, page 106
- **Using the HTML Template Wizard**, page 109
- **Customizing HTML Before Generation**, page 123

Introduction

An HTML template is a file that defines a web page or a component of a web page. HTML templates are generated using the HTML Template wizard, which uses information stored in an ABO to select the content and layout of either a maintenance or browse page. Besides HTML code, templates are generated with JavaScript to perform some processing and links to cascading style sheets to give a consistent appearance to pages throughout the application.

Framework HTML Templates

Besides the HTML templates that you generate, your web pages use framework 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 your pages, depending on whether the page presents maintenance or browse functionality.

For a full list of the framework templates and information about how the framework and generated HTML templates work together to create a page, see **Framework HTML Templates and Page Handlers**, page 77.

Replacement HTML Tags

Templates are also generated with Construct Spectrum replacement HTML tags to act as placeholders for other templates and active content 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 have been 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, you can modify the specifications for some HTML tags, effectively customizing the template before you generate it. **Customizing HTML Before Generation**, page 123, explains how to use the **Customize HTML** dialog to do this.

After Generating the Template

After you have generated a template, you can customize the HTML in any way you wish. However, if you need to regenerate the template using the wizard, your changes will be lost. Consequently, you may wish to rename your HTML template, generate a new template, and copy your customizations into the new template. If you have a code comparison utility installed and configured for use with Construct Spectrum, you may find it useful for comparing regenerated with original HTML. For more information, see **Features of the ABO and Web Wizards**, page 59, *Construct Spectrum Programmer's Guide*.

Using Replacement Tags

Another way to customize your web pages is to create your own replacement HTML 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. This custom code can also be protected using the PROTECT tag. For information about creating custom tags, see **Construct Spectrum Replacement HTML Tags**, page 145.

Using Framework Components

If you wish to perform customizations on web pages throughout an application, you can modify the framework 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 more information, see **Framework Components Supplied with Construct Spectrum**, page 71.

Another alternative is to incorporate customizations into the wizards themselves so that they are generated into the template. To do so, you can modify the code frames supplied with Construct Spectrum.

Using the HTML Template Wizard

Using the HTML Template wizard, you can generate templates for each of the ABOs in your application, resulting in a web page for each business object.

For information about creating ABOs, see **Using ActiveX Business Objects**, page 107, *Construct Spectrum Programmer's Guide*.

Using the HTML template wizard to create a template involves the following steps:

- Invoking the wizard
- Selecting an ABO
- Confirming the ABO's details
- Configuring the template
- Customizing the HTML that will be generated (optional). For information about this step, see **Customizing HTML Before Generation**, page 123.
- Generating the template and saving it to your Visual Basic project, with the optional steps of viewing the generation report, comparing code, and previewing the web page.

The following sections explain the steps in detail.

Invoking the Wizard

The HTML Template wizard is available as 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 and select the web project in Project Explorer.
 - 2 From the **Spectrum** menu, point to **Wizards** and select **HTML Template**. The HTML Template wizard appears:



HTML Template Wizard

For information about using the Spectrum Cache viewer  or Configuration editor , see **Features of the ABO and Web Wizards**, page 59, *Construct Spectrum Programmer's Guide*.

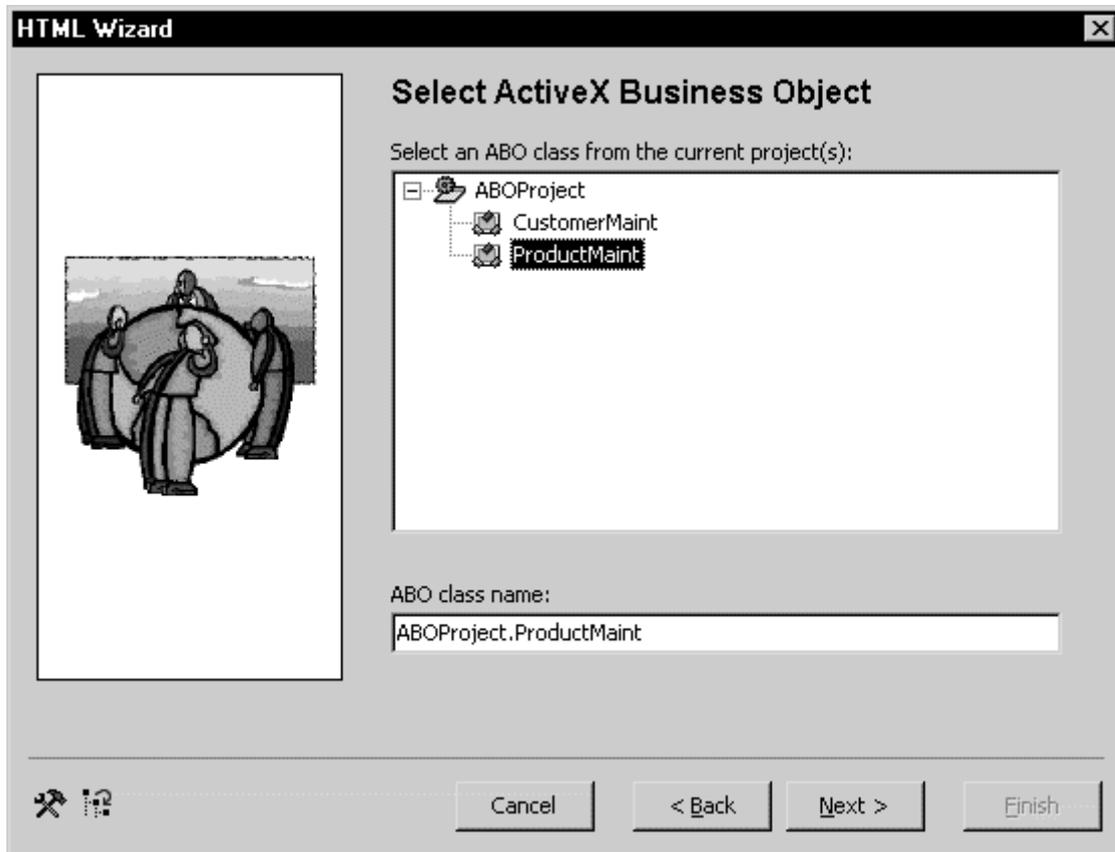
Selecting an ABO

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

➤ To select an ABO:

1 Click **Next**.

The **Select ActiveX Business Object** step appears:

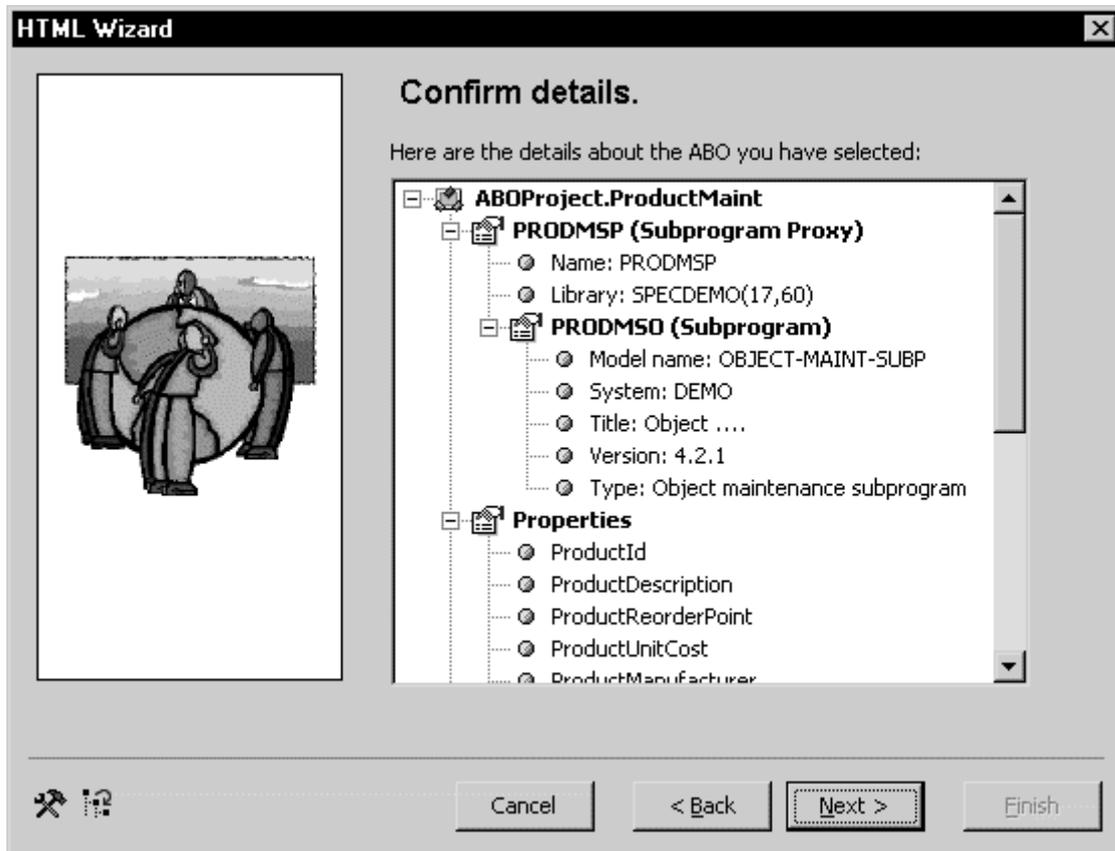


HTML Template Wizard — Select ActiveX Business Object

- 2 Select an ABO from the ABO project in the group currently open in Visual Basic.
- 3 Click **Next**.

Confirming ABO Details

When you select an ABO and click **Next**, the wizard presents a summary of the ABO:



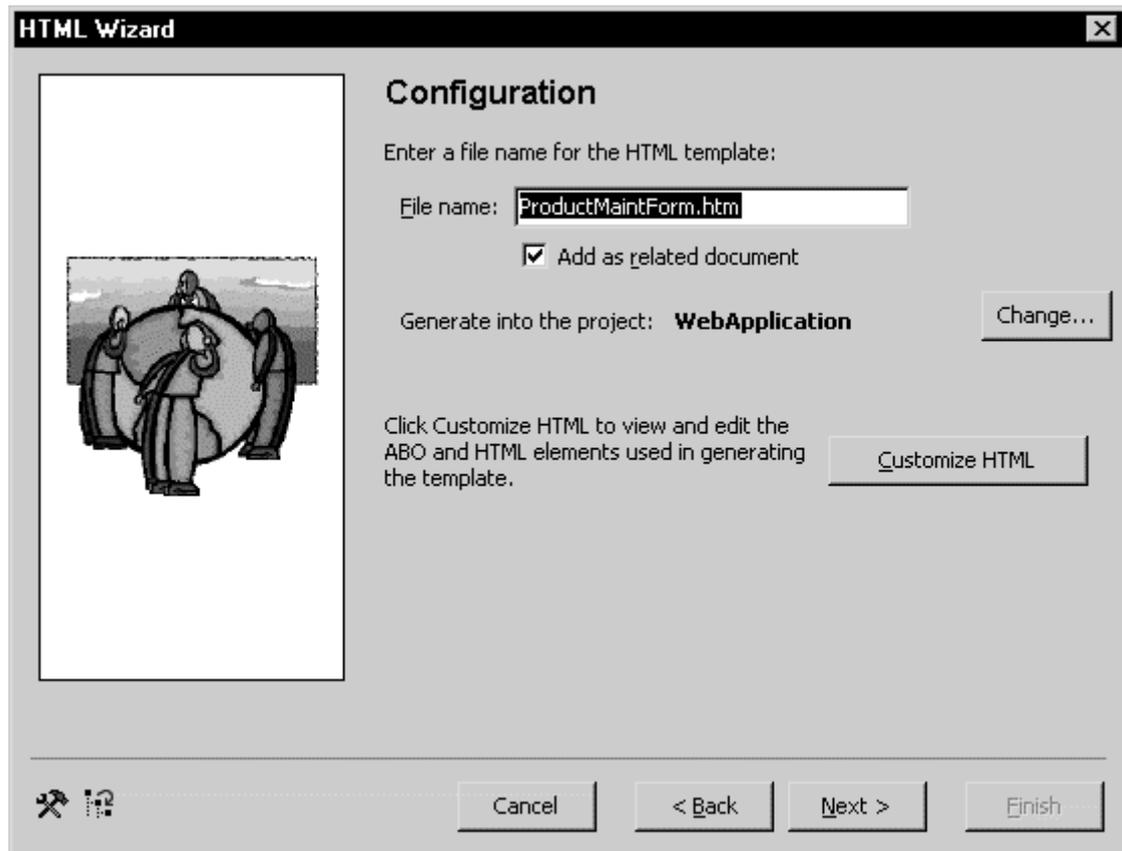
HTML Template Wizard — Confirm Details

The list displays the subprogram proxy and subprogram that the ABO encapsulates, as well as the properties associated with it.

- At this point, you can either:
- Click **Back** to select another ABO.
- Or
- Click **Next** to proceed to the next step.
The wizard performs the following steps:
 - Reads the ABO specification to confirm that the functionality it requires is supported. If the functionality isn't supported, a message appears prompting you to select a different ABO class.
 - Reads the field names and properties exposed by the ABO.

Configuring the HTML Template

When you select an ABO and click **Next** to confirm your selection, the wizard presents configuration options:



HTML Template Wizard — Configuration

- To configure the HTML template:
 - 1 Check the file name and change it, if necessary.

You can change the file name under which the HTML template will be saved by typing the new name in the **File name** text box. The HTML file will be saved in the same directory as your web application.

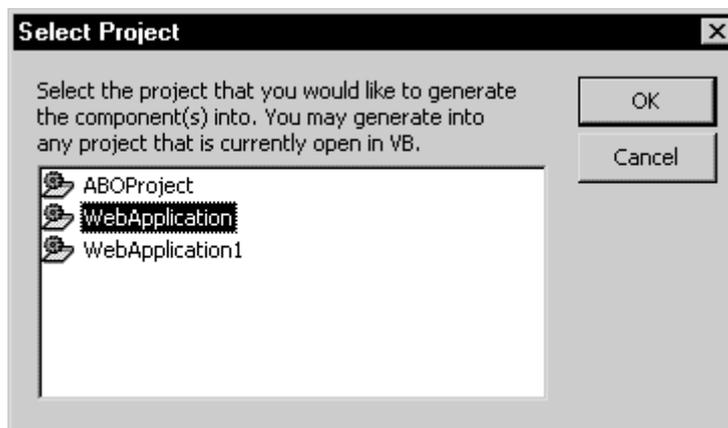
Note: If you change the file name for the HTML template, be sure to change the page handler's reference to the name.

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 change how default names are derived for HTML templates using the Configuration editor. For more information, **The Configuration Editor**, page 60, *Construct Spectrum Programmer's Guide*.

- 2 Check that the correct web project is selected and change it, if necessary.

By default, the template is saved to the web project that you selected in the Project Explorer before invoking the wizard. However, if you would like to save the file to another project, click **Change** to locate and select another project. The **Select Project** window appears:



Select Project

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

- 3 If you wish 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 double-click an HTML file in Project Explorer to open it, your web browser will open, rather than your HTML editor. To edit the HTML, either associate the htm file type with your HTML editor or open the file from within the editor.

- 4 If you wish to view and edit the fields that will appear on your web page, click **Customize HTML**.

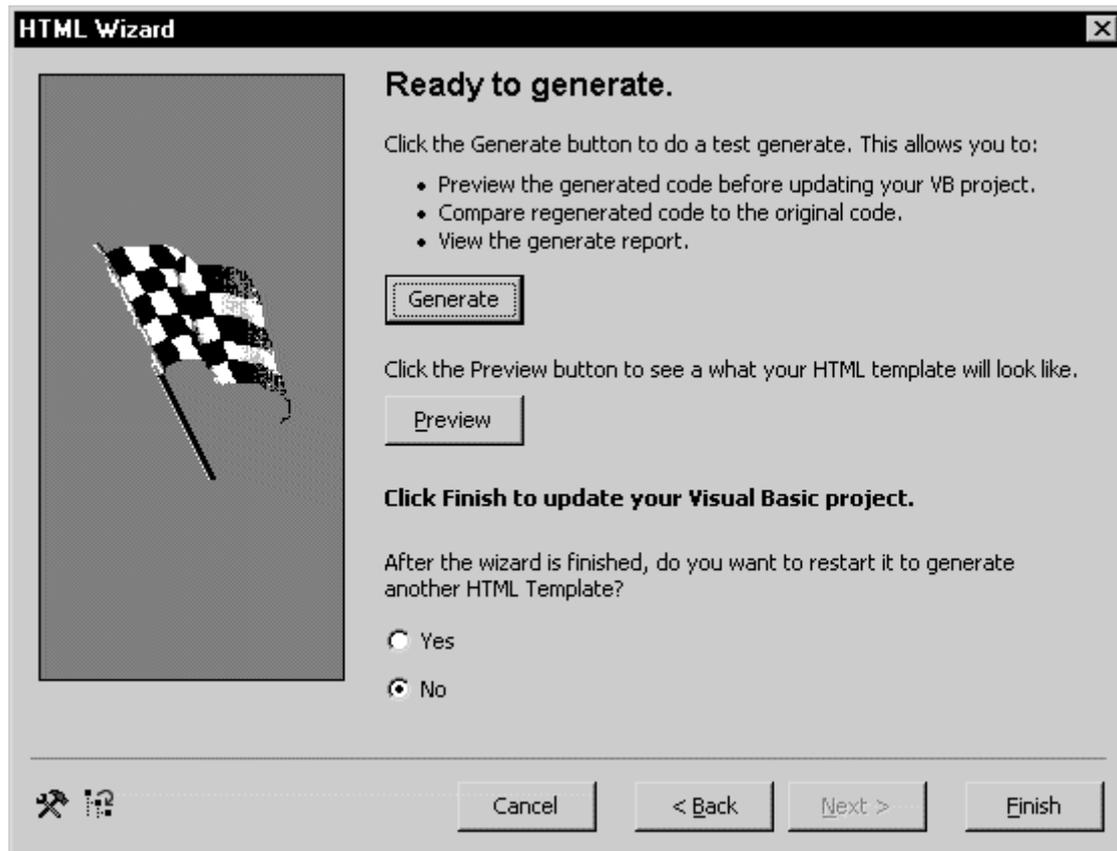
The **Customize HTML** dialog appears, showing the ABO properties (fields) that will be generated in your HTML template. When you have finished customizing the HTML, click **Close** to return to the **Configuration** step.

For information about this dialog, see **Customizing HTML Before Generation**, page 123.

- 5 Click **Next**.

Generating the HTML Template

When you click **Next** after configuring your HTML template, the wizard presents generation options:



HTML Template Wizard — Ready to Generate

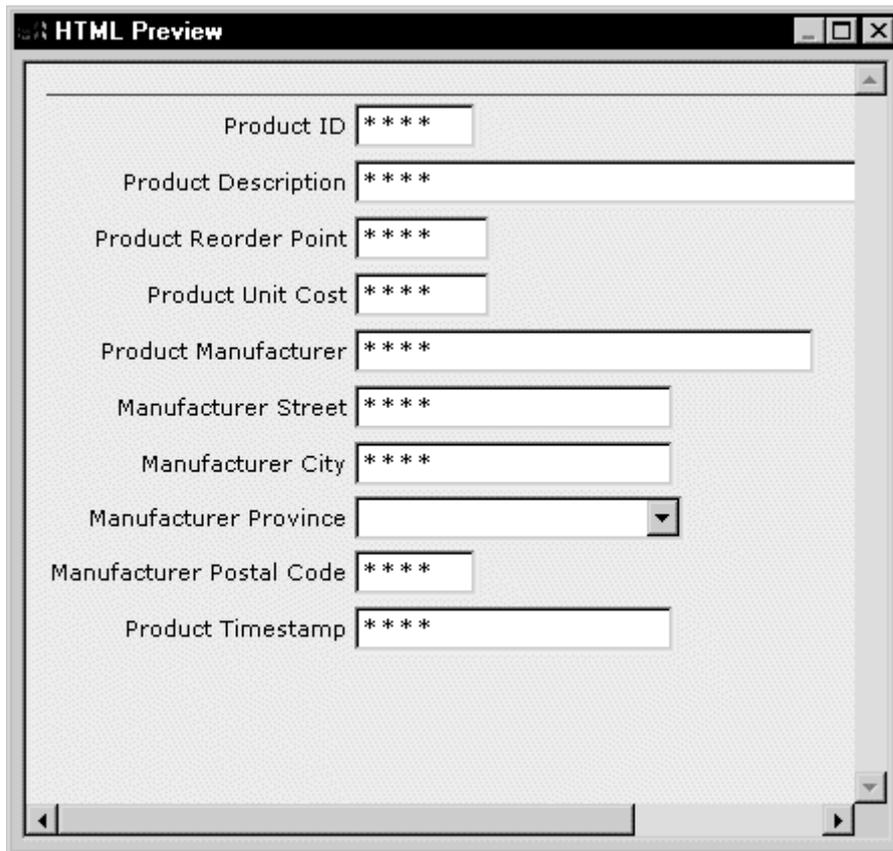
➤ At this point, you can perform four actions:

- 1 If you wish 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 you generated with code from an earlier generation of the module.

For information about using a code comparison utility with Construct Spectrum, see **Using Reports with a Code Comparison Tool**, page 82, *Construct Spectrum Programmer's Guide*.

For information about the generate report, see **Using Reports**, page 77, *Construct Spectrum Programmer's Guide*.

- 2 If you wish to view your HTML code as it will appear in a web browser, click **Preview**. For example:



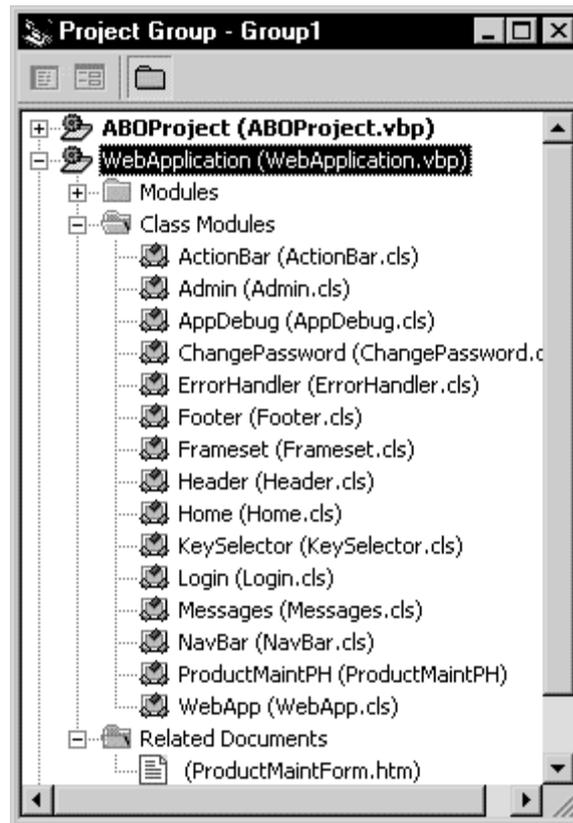
Preview of HTML Code

- 3 If you wish to invoke the wizard again to generate another HTML template, select **Yes**.
- 4 When you wish to generate the code, update the Visual Basic project, and close the wizard, click **Finish**.

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

After Generation is Complete

As a result of generating the HTML template, the new HTML template 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. For example:



Project Explorer — Web Project after Generating an HTML Template

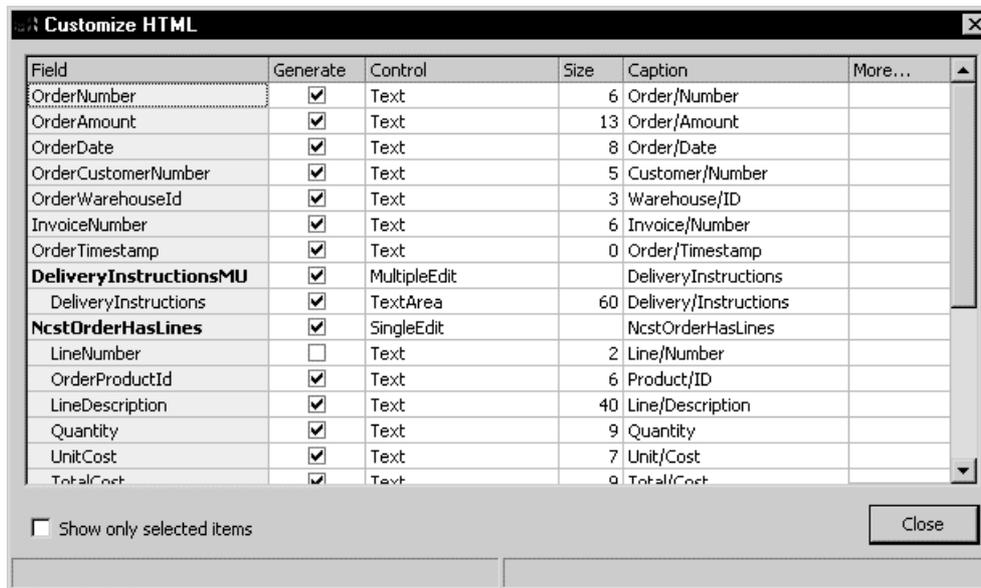
The next step after creating an HTML template is to update the project's object factory. You can then run the application, test, debug, and customize it.

For more information, see **Updating and Customizing the Object Factory**, page 163.

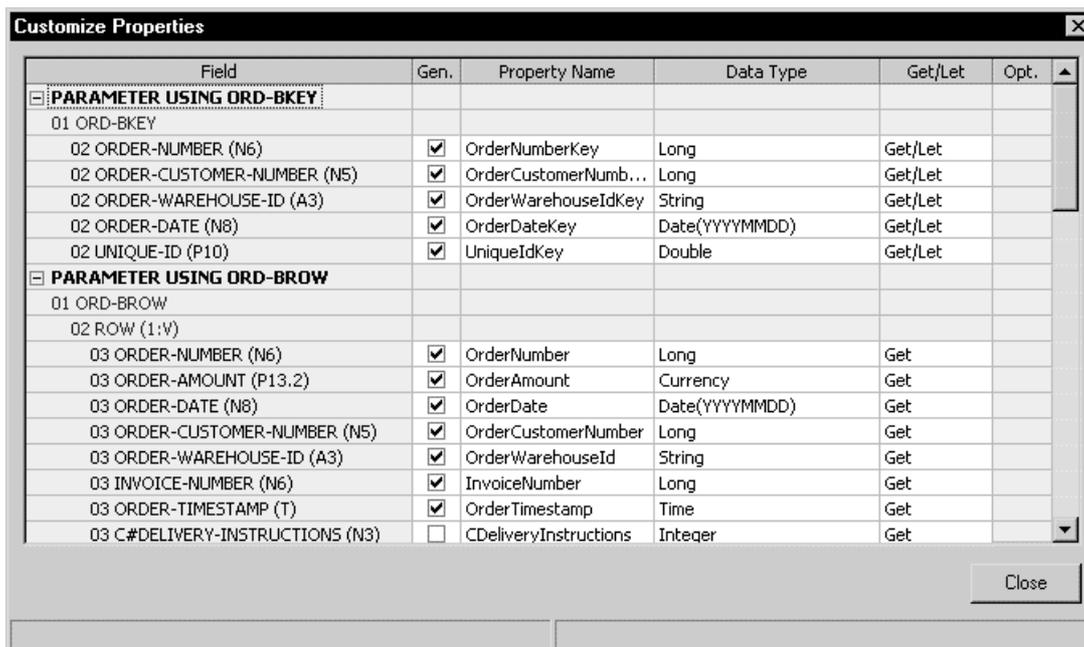
Customizing HTML Before Generation

If you click **Customize HTML** on the Configuration step of the HTML Template wizard, the **Customize HTML** dialog appears. This dialog displays all of the database fields that will be included in your HTML page, as well as their characteristics.

The dialog differs depending on whether you are generating a template for a maintenance or browse page. Here are examples of both:



Customize HTML for a Maintenance Template



Customize HTML for a Browse Template

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 appears in the message area at the bottom of the dialog.

The following sections explain how to customize maintenance and browse templates.

Customizing Maintenance Pages

You can perform the following customizations to the HTML for maintenance pages using the **Customize HTML** dialog:

- 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

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

Deselecting 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, click its **Generate** check box to deselect it.

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

Changing the Type of Control for a Field

This section explains how controls are derived from field properties in Predict and how to change the control for a field.

How Controls are Derived

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 value (MU) fields, periodic groups (PE), and related files are generated as HTML tables called “sections” in Construct Spectrum web applications. MU fields are generated with the multiple edit view. PEs and related files are generated with the single edit view. For more information, see **Changing View Options for Sections**, page 131.
- 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.

Controls in Construct Spectrum Web Applications

The following table summarizes the controls that are derived:

Predict Equivalent	HTML Equivalent	Tag example	Visual Basic 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

Predict Equivalent	HTML Equivalent	Tag example	Visual Basic GUI Equivalent
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

Changing the Control for a Field

- To change the default control for a field:
 - 1 Click its Control cell to open the drop-down list.
 - 2 Open the list and 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, as explained in the next sections.

Specifying or Modifying Options in a Selection List

There are two ways that a field can be represented by a selection list in the HTML template:

- The field has a table verification rule attached to it in Predict. In this case, you can view the rule and modify the options that will appear in the list.

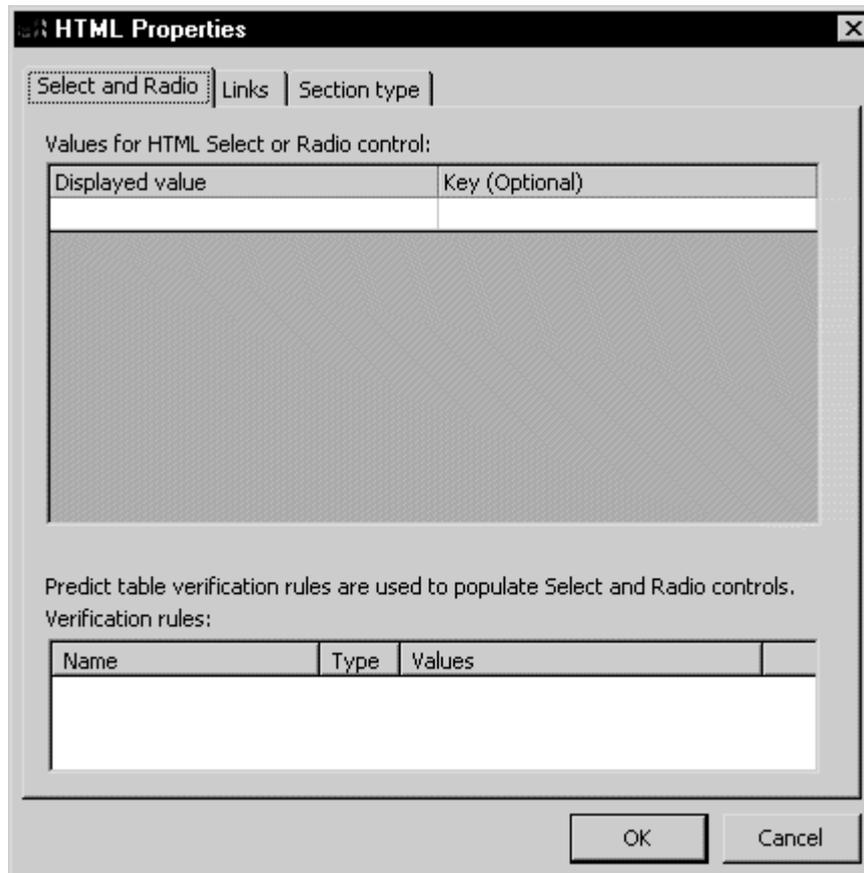
Or

- The field has the GUI_COMBOBOX keyword attached to it in Predict. In this case, you can specify the options to appear in the list.

Or

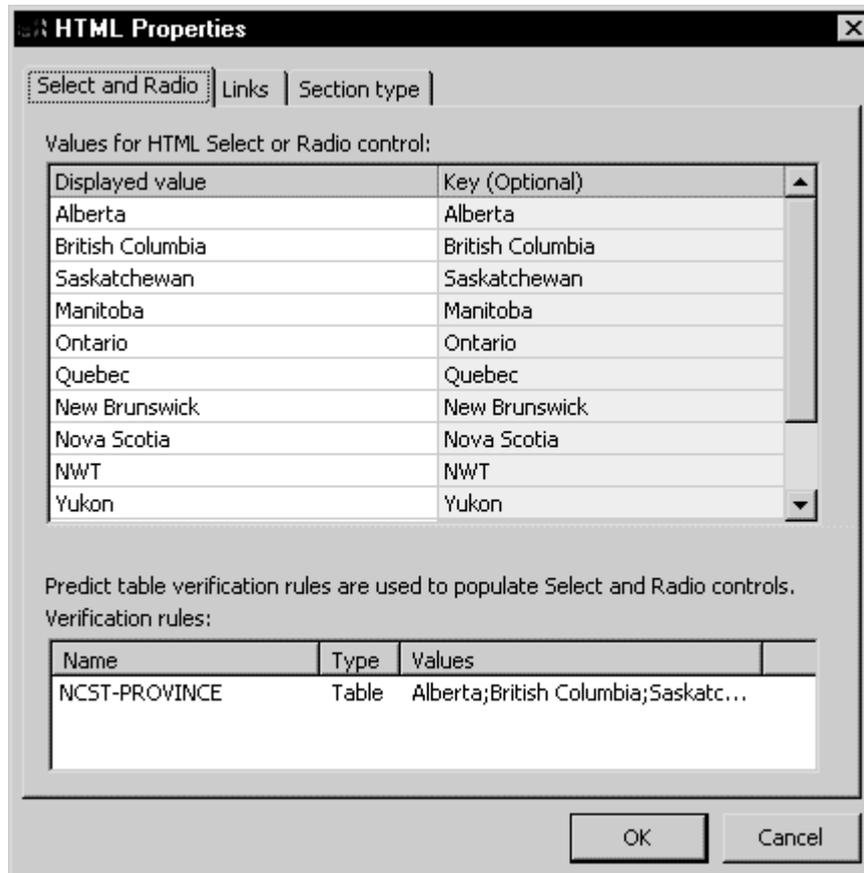
- You changed the field's control to Select. In this case, you can specify the options to appear in the list.

- To specify or modify options in a selection list:
- 1 Click the field's More cell to make the button visible and click it. The **HTML Properties** dialog appears.
 - 2 Click the **Select and Radio** tab:



HTML Properties — Select and Radio

If the field has a table verification rule attached to it in Predict, the rule and its values are displayed in the dialog. For example:



HTML Properties — Element Properties Showing Verification Rule

Tip: You can use the **Select and Radio** tab to view verification rules attached to any field in the template, not just selection lists. The tab shows all verification rules attached to a field.

- 3 Take one of the following actions:
 - If a table verification rule is shown in the dialog, you can edit the options in the **Displayed value** column. However, you cannot remove them or specify additional values. For example, a verification rule might define state names using abbreviations. You could replace the abbreviations with full names for display in the selection list on the web page.
- Or
- 1 If you have 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 wish to display in the **Displayed value** column. Press the Tab key to create new rows in the table.
 - 2 If the displayed values must be changed before they can be returned to the database, specify those values in the **Key (Optional)** column.
 - 3 Click **OK** to save your changes and close the **HTML Properties** dialog.

Specifying or Modifying Options in a Radio Button Group

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

- The field has the GUI_OPTIONBUTTON keyword attached to it in Predict. In this case, you can specify the radio buttons to appear in the group.
- Or
- The field has a table verification rule and the GUI-OPTIONBUTTON keyword attached to it in Predict. In this case, you can view the rule and modify the radio buttons that will appear in the group.
- Or
- You changed the field's control to Radio. In this case, you can specify the radio buttons to appear in the group.

- To specify or modify radio buttons in a group:
- 1 Click the field's More cell to make the button visible and click it. The **HTML Properties** dialog appears.
 - 2 Click the **Select and Radio** tab.

If the field has a table verification rule attached to it in Predict, the rule and its values are displayed in the dialog. For an example, see **Specifying or Modifying Options in a Selection List**, page 127.

Tip: You can use the **Select and Radio** tab to view verification rules attached to any field in the template, not just radio button groups. The tab shows all verification rules attached to a field.

- 3 Take one of the following actions:
 - If a table verification rule is shown in the dialog, you can edit the displayed values. However, you cannot remove them or specify additional values. For example, a verification rule might define Credit Ratings using abbreviations. You could replace the abbreviations with full names for display in the radio button group on the web page.
- Or
- 1 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 wish to display in the **Displayed value** column.
 - 2 If the displayed options must be changed before they can be returned to the database, specify those values in the **Key (Optional)** column.
 - 3 Click **OK** to save your changes and close the **HTML Properties** dialog.

Changing View Options for Sections

Period groups (PE), multi-value fields (MU), and related files are generated as sections in your 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 the alpha datatype, you have an additional option: multiple edit text area.

Single Edit View

By default, sections use single edit view, as in the following example:

Line Distribution View

Index 1 < > Add Insert Delete Clear

Cost Center 15

Acct 9342

Project AV

Dist Amount \$5,000.00

Single Edit View

Users can view one record at a time and edit all fields in the record.

Single Edit View with Report View Option

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

Line Distribution View				
#	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:

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

Multiple Edit View

Users can edit any field.

Multiple Edit Text Area View

The section is presented as a text area in which users can add and edit text:

DeliveryInstructions ▾
<input type="text"/>

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

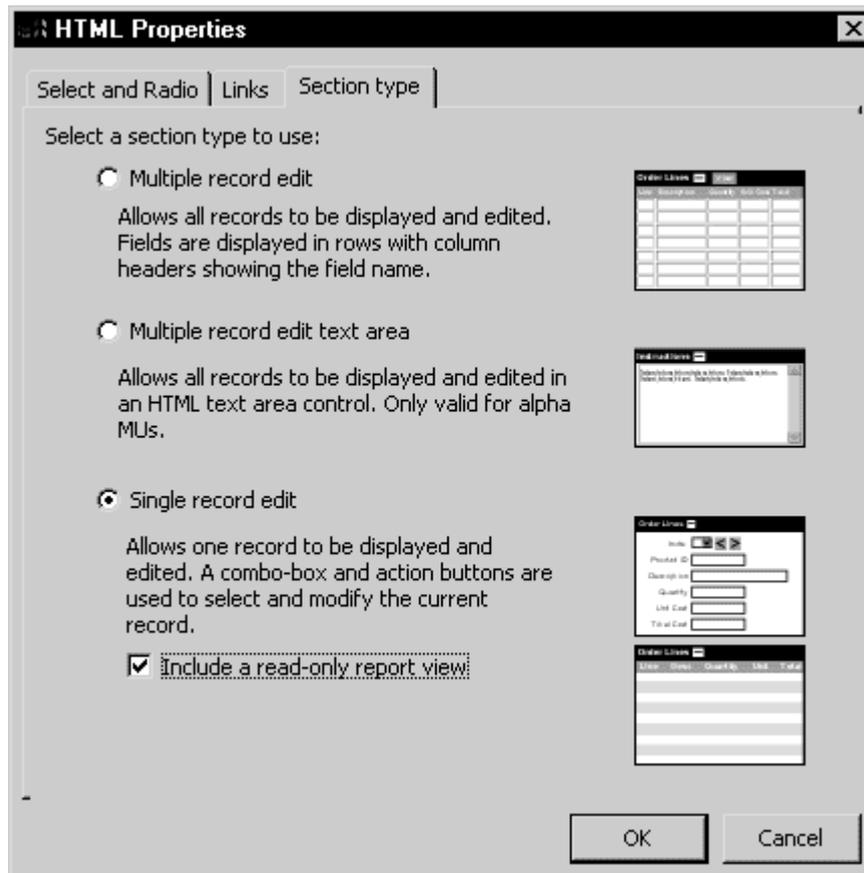
When viewed with Internet Explorer, sections are collapsible. Users can close and reopen sections using the minus (-) and plus (+) icons, respectively. This functionality saves space on the page and reduces the need for scrolling. However, Navigator does not support collapsible sections. If your application will be used with Navigator, you may consider using only single edit view or single edit view with the report view option to limit the length of the page.

Changing View Options

- There are two ways to change a section's view options:
- 1 Click the section's Control cell to open the drop-down list.
 - 2 Select an option: Single Edit, Single Edit and Report, Multiple Edit, or Multiple Edit Text Area.

Or

- 1 Click in the section's More cell to enable the button and click it. The **HTML Properties** dialog appears.
- 2 Click the **Section type** tab:



HTML Properties — Section Type

- 3 Select an option.
- 4 Click **OK** to save your change and close the dialog.

Changing the Width of a Text Box or Text Area

The default width for a control appears 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, 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.

Changing the Control's Caption

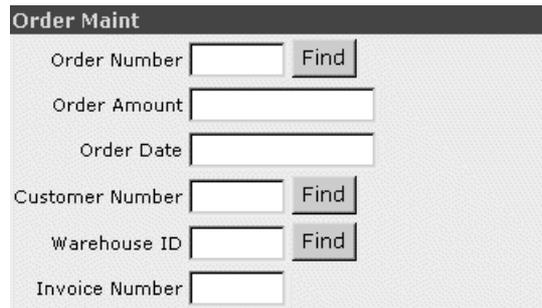
The default caption for each field is determined by the field name stored in Predict or the name as modified in the ABO. You can change the caption for the field's control.

- To change the text, replace the value in the **Caption or Header** column.

Creating a Link to a Browse Page

If your application includes an ABO and page handler for a browse object, you can create a link on the maintenance page to a corresponding field in a browse page. Users click the **Find** button next to the field to open the browse 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.

For example, if you are creating a maintenance page for an Order object, one of the fields on the page might 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 the Warehouse ID field:



The screenshot shows a web form titled "Order Maint". It contains several input fields and buttons:

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

Order Maintenance

The user can click the link to invoke the Warehouse browse page:



The screenshot shows a web page titled "Warehouse Browse". It features a search interface and a table of warehouse records.

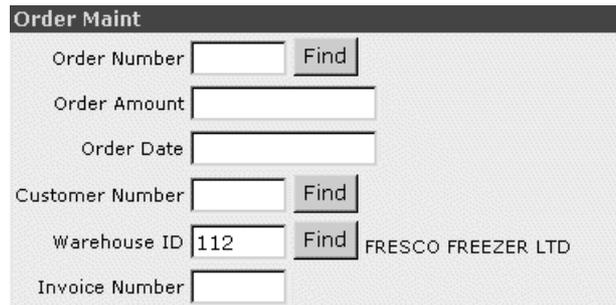
Search Interface:

- Sort Key: WarehouseId
- WarehouseIdKey:
- Range: *
- Go button

#	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

By clicking the red arrow next to a Warehouse ID, the user switches back to the maintenance page, which shows the selected Warehouse ID in the original field:



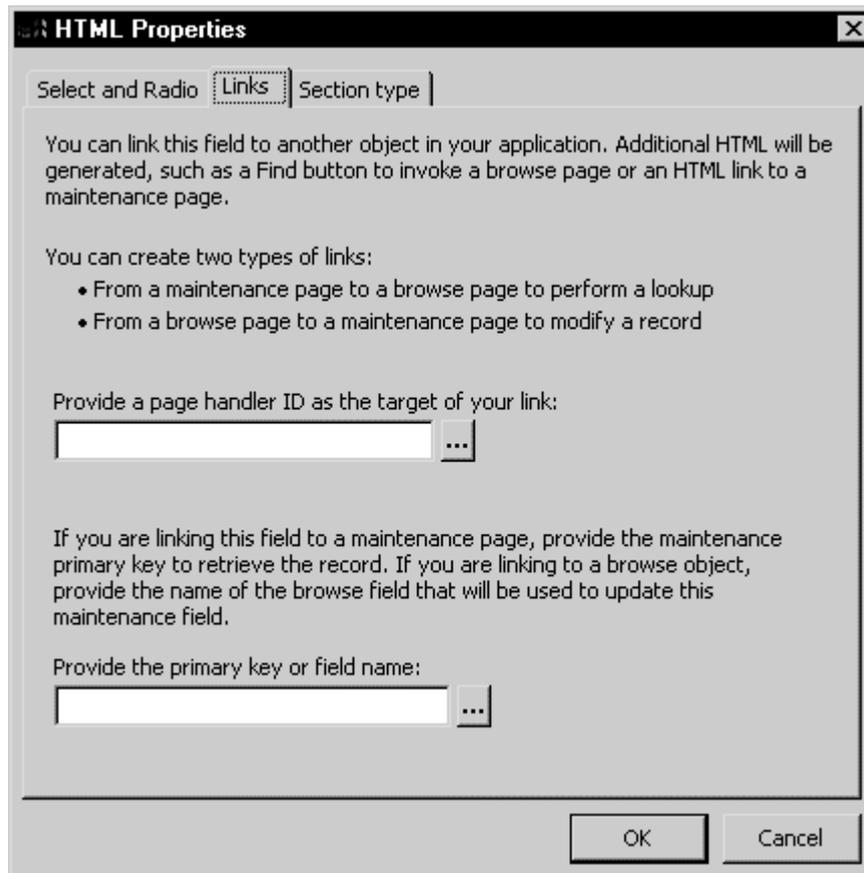
The screenshot shows a web form titled "Order Maint" with a dark header. The form contains several input fields and buttons:

- Order Number: Find
- Order Amount:
- Order Date:
- Customer Number: Find
- Warehouse ID: Find FRESKO FREEZER LTD
- Invoice Number:

Customer Maintenance after Look Up

Creating a Link to a Browse Page

- To create a link from a maintenance field to a browse page:
 - 1 Click the field's More cell to enable the button and click it. The **HTML Properties** dialog appears.
 - 2 Click the **Links** tab:



HTML Properties — Links

- 3 Provide the page handler ID of the browse page that will be the target of your link.
- 4 Provide the name of the browse field that will be the target of your link.
- 5 Click **OK** to save your changes and close the **HTML Properties** dialog.

The return link from the browse to the maintenance page is supplied automatically by the HTML Template wizard.

Customizing Browse Pages

You can perform the following customizations to the HTML for browse pages using the **Customize HTML** dialog:

- Deselect fields for generation
- Change the alignment of columns in the browse
- Add links from browse fields to maintenance pages
- Change the headers for browse columns

The following sections explain how to perform these customizations.

Deselecting 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, click its **Generate** check box to deselect it.

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

Changing the Alignment of a Column in a Browse Table

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

Adding 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 in the field on the browse page to the maintenance page. The user can click the link on a particular record in the browse page to open the maintenance page, where the selected record is displayed. The user can then edit the record in the maintenance page.

For example, if you are creating a browse page for a Customer object, you can specify that records on the browse page be links to the Customer maintenance page. The following illustration shows an excerpt from the Customer browse page in the Demo web application. The Customer Number and Warehouse ID values are links:

Customer2 Browse

Sort Key: CustomerNumber

CustomerNumberKey:

Range: *

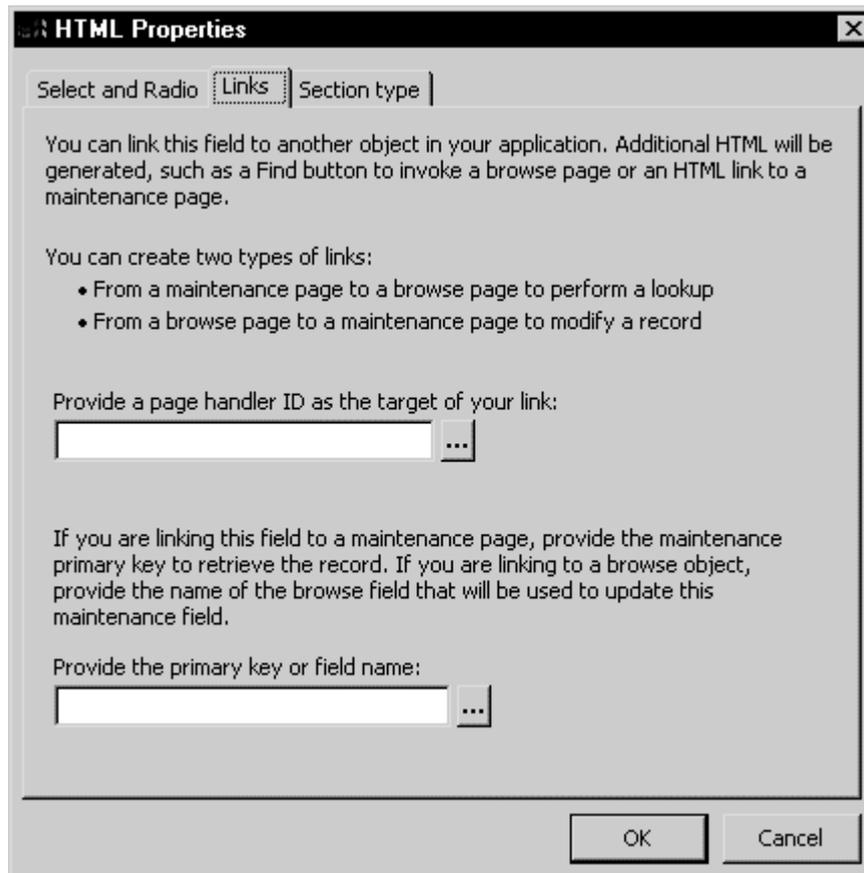
Go More Display all fetched rows

#	Customer Number	Business Name	Phone Number	Credit Rating	Credit Limit	Discount %	Warehouse ID
1	1	SAGA SOFTWARE INC (CANADA)	(519) 622-0889 C		\$2,000.00	\$5.00	524
2	2	JOURNEYMEN FABRICATING	(519) 234-6422 z		\$500,000.00	\$12.00	524

Customer Browse

Creating a Link to a Maintenance Page

- To create a link from a browse field to a maintenance page:
 - 1 Click the field's More cell to enable the button and click it. The **HTML Properties** dialog appears.
 - 2 Click the **Links** tab:



HTML Properties — Links Tab

- 3 Provide the page handler ID of the maintenance page that will be the target of the link.
- 4 Provide the primary key that will retrieve the record in the maintenance page.
- 5 Click **OK** to save your changes and close the **HTML Properties** dialog.

Changing 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, replace the value in the **Caption or Header** column.

CONSTRUCT SPECTRUM REPLACEMENT HTML TAGS

Construct Spectrum's replacement HTML tags allows you to have "live" content presented programmatically on your web pages. Special replacement indicator tags are embedded in HTML templates. This chapter explains the syntax of the tags, the tags supplied with Construct Spectrum, and how to create your own tags.

The following topics are covered:

- **How Page Handlers Process Tags**, page 146
- **Syntax of Replacement Tags**, page 147
- **Types of Replacement Tags**, page 148
- **Replacement Tags Supplied with Construct Spectrum**, page 150
- **Defining Custom HTML Replacement Tags**, page 160

How Page Handlers Process Tags

Page handlers interact with the ABO and the HTML templates to gather the information they need to assemble appropriate content.

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, scanning for replacement tags.
- For each tag read, the page handler:
 - calls the tag processing framework component for custom tags
 - does specific processing within the template loop
 - does replacements that are global to the application.
- Replaces the tag with appropriate content.
- Sends content to the browser once all the replacements have been made.

The prefix signals the page handler that it must find appropriate content to display in the web page that has been requested. For example, consider what happens when the page handler reads the following replacement tag:

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

In this example, the page handler “knows” that it must create 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 in the buffer, too. 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 of Replacement Tags

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

Construct Spectrum replacement tags follow this syntax:

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

All Construct Spectrum replacement tags use upper case for tag names. Tag names are case-sensitive; for example, FIELD is a different tag name than Field. Attribute information is not case-sensitive.

Types of Replacement Tags

While all Construct Spectrum replacement tags follow the syntax explained in the previous section, they differ in complexity. These variations were developed to address typical replacement needs.

Replacement tags can be categorized according to these types:

- Simple
- Conditional
- Repeating
- Complex

Simple

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

Conditional

Conditional replacement tags require checking to determine whether to display the requested value. For example, the SECURITY tag checks whether the user making the request has permission to retrieve the information requested. If the user is permitted access to the requested information, the requested content is displayed in the browser; otherwise, the content returned is set to display as blank lines.

Repeating

Repeating replacement tags use loop processing. Such tags describe the appearance of one row, specifying the field name to use, the attributes for the field, and the number of times the process is to be performed. Substitutions are made for each repetition.

The following example implements repeating replacement tag processing:

```
<!--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's 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. Complex replacement tags can combine, nest, or embed tags within the single structure. These tags can 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 will find multiple values for Customer Number. Once it has stored the customer number for the number of rows specified, all the customer numbers are sent to the browser at once.

Note: While all tags can be nested, place your end tags carefully. Overlapping tags result in XML formation errors.

Replacement Tags Supplied with Construct Spectrum

This section explains the function and attributes of the tags and gives examples of each.

ALTERNATE

This tag is usually contained inside a REPEAT tag. It varies a value based on a counter. For example, if the Freq attribute is set to 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 specific level. The Type attribute is used to indicate 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 an input tag of type "CHECKBOX." The Value attribute of the input tag returns an "X" if the user turns on the checkbox.

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-->
```

ERROR

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

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 on.

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-->
```

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-->
```

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"/-->
```

RADIO

This tag adds the CHECKED tag to the correct input tag of type "RADIO."

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 that contains the number of repetitions. By default, the REPEAT tag replaces all occurrences of %1 with the current repeat index. Use of the Field attribute repeats 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 supplied in the Tag attribute. There are three types of security tag: 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 field 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** would be inserted in the option tag that contains the Value **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>
```

SUBMIT

This tag returns a URL to navigate to a specific page handler. The page handler will process the action contained in the action attribute, if included.

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 for the current page handler.

Example of the TITLE tag

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

Defining Custom HTML Replacement Tags

You can customize your application with custom replacement tags to override supplied ones. Creating your own replacement tags lets you add functionality without affecting supplied replacement tags.

There are two steps to creating custom replacement tags:

- 1 Add your custom tags to the HTML template where you want content to be replaced dynamically.
- 2 Modify either the page handler's ParseTemplate function (if you want the tags to be used in one page only) or add code to the TagProcessing.bas file (if you want the tags to be used throughout the web application).

Modifying TagProcessing.bas

The TagProcessing.bas module is one of the frameworks components that is added to your web project. It is stored with the web application's modules. Add your custom replacement tags to the TagProcessing.bas module if you wish to use them throughout the web application.

The TagProcessing.bas module contains user-defined tag overrides and customizations.

Syntax for the 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
```

Modifying the Page Handler

The following example is from the ParseTemplate function of a maintenance page handler. A custom tag exit has been added.

```
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
        End Select
    End If
End Function
```

```
'<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 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.

The following topics are covered:

- **Introduction**, page 164
- **Using the Object Factory Wizard**, page 165
- **User Exits in the Object Factory**, page 170

Introduction

Each Construct Spectrum web application contains a module called the object factory (Ofactory.bas). The object factory encapsulates all the business objects in the application, making it aware of the objects and the actions, such as maintenance and browse actions, enabled for the objects. Whenever you add or modify an ABO or a page handler, you must update the application's object factory.

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 more information about using the object factory to modify security, see **Securing Your Application**, page 179.

Using the Object Factory Wizard

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

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

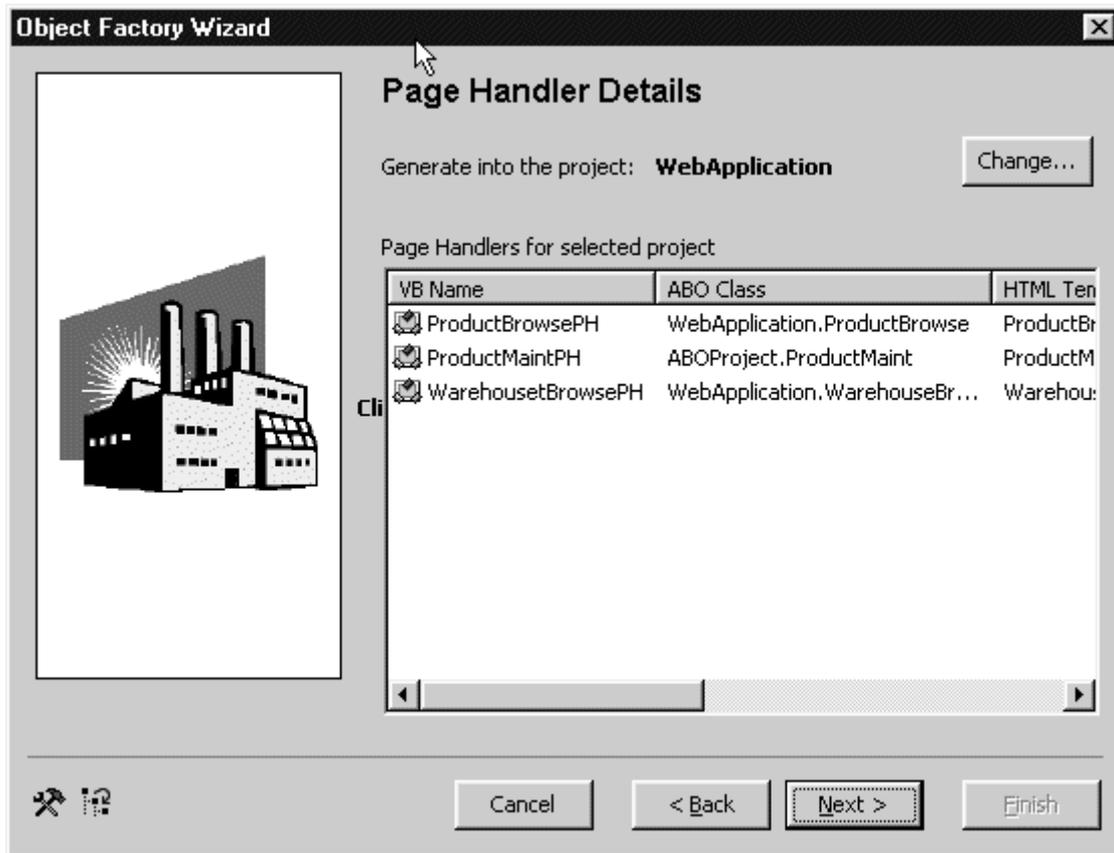
- To update your object factory:
 - 1 Open the project group for your application in Visual Basic and select the web project in Project Explorer.
 - 2 From the **Spectrum** menu, point to **Wizards** and then select **Object Factory**. The Object Factory wizard appears:



Object Factory Wizard

For information about using the Spectrum Cache viewer  or Configuration editor , see **Features of the ABO and Web Wizards**, page 59, *Construct Spectrum Programmer's Guide*.

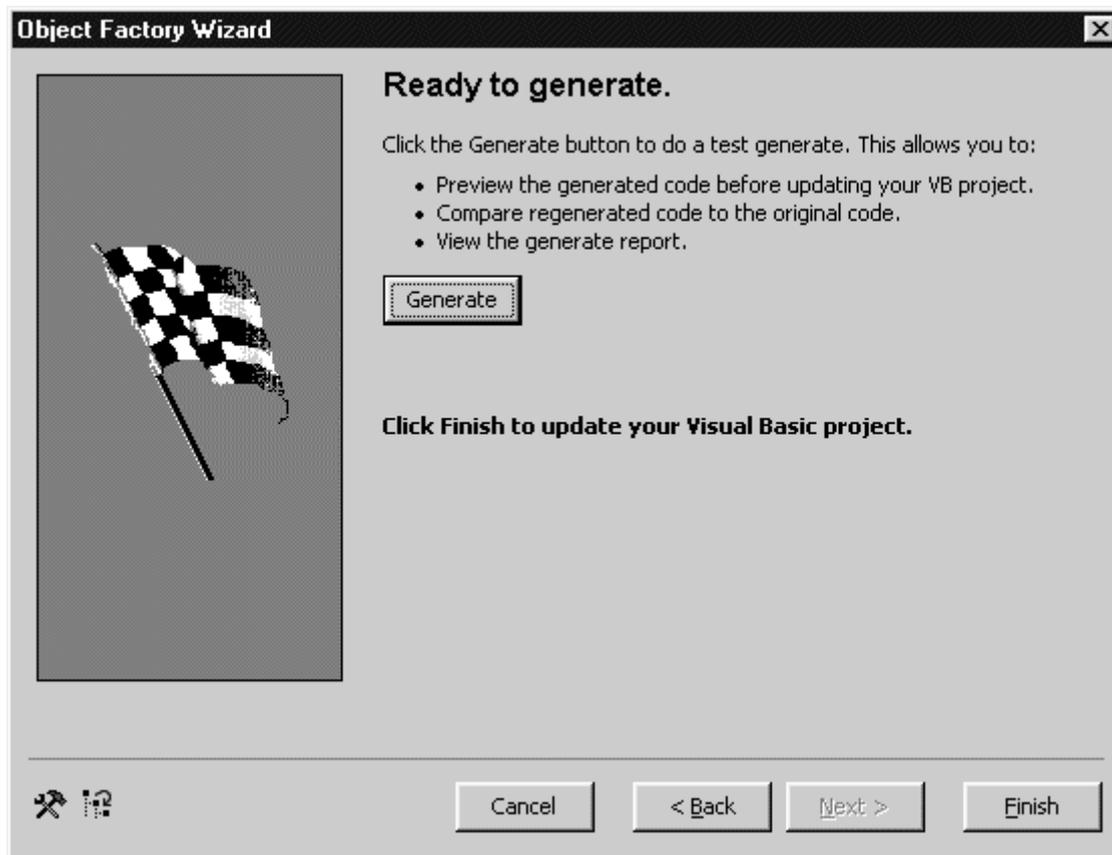
- 3 Click **Next**.
The **Page Handler Details** step appears:



Object Factory Wizard — Page Handler Details

- 4 Check that the correct web project is selected and change it, if necessary, by clicking **Change** and selecting another project.
- 5 Select a page handler.

- 6 Click **Next**.
The **Ready to Generate** step appears:



Object Factory Wizard — Ready to Generate

7 At this point, you can perform two actions:

- If you wish 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 you generated with code from an earlier generation of the module.

For information about using a code comparison utility with Construct Spectrum, see **Using Reports with a Code Comparison Tool**, page 82, *Construct Spectrum Programmer's Guide*.

For information about the generate report, see **Using Reports**, page 77, *Construct Spectrum Programmer's Guide*.

- When you wish to generate the code, update the Visual Basic project, and close the wizard, click **Finish**.

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

User Exits in the Object Factory

The object factory contains the `DefaultPage.SetDefault` user exit and three user exits that 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 more information about coding user exits for security purposes, see **Securing Your Application**, page 179.

VALIDATING YOUR DATA

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

The following topics are covered:

- **Types of Validations Used in Web Applications**, page 172
- **How Errors are Displayed in Web Pages**, page 174
- **The Debug Page**, page 177

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 validation rules derived from the Natural subprogram

The following sections explain these types of validations in detail.

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 could 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 dialog. 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 more information about using and creating BDTs see, **Using Business Data Types**, page 153, *Construct Spectrum Programmer's Guide*.

For information about using the `ICSTPageHandler_BDTOverrides` user exit to modify BDTs, see **User Exits in Page Handlers**, page 97.

Validations in the ABO

You can use user exits to code validations in the ABO. For example, you can provide code to validate a value before the Property Let function for any property in the ABO. For information about using implied user exits in the ABO, see **Working with Code**, page 70, *Construct Spectrum Programmer's Guide*.

The advantage of adding validation code to the ABO rather than to the page handler is that the validations are available to any other page handlers you may create based on the ABO.

For more information, see **Using ActiveX Business Objects**, page 107, *Construct Spectrum Programmer's Guide*.

Validations in the Page Handler

The page handler provides a Client Validations user exit, in which you can code validation routines that are specific to a web page.

For more information, see **PerformAction.ClientValidations**, page 100.

How Errors are Displayed in Web Pages

This section describes how errors are presented in a web application.

How Errors Are Displayed in Internet Explorer

When an error occurs in a Construct Spectrum web application accessed with Internet Explorer, the pertinent text boxes are highlighted and an explanation of the error appears in the message area at the bottom of the page. To resolve the error, the user must clear the text box or provide correct information.

For example, the following illustration from the Demo web application shows an excerpt from the Customer Maintenance page with two BDT errors.

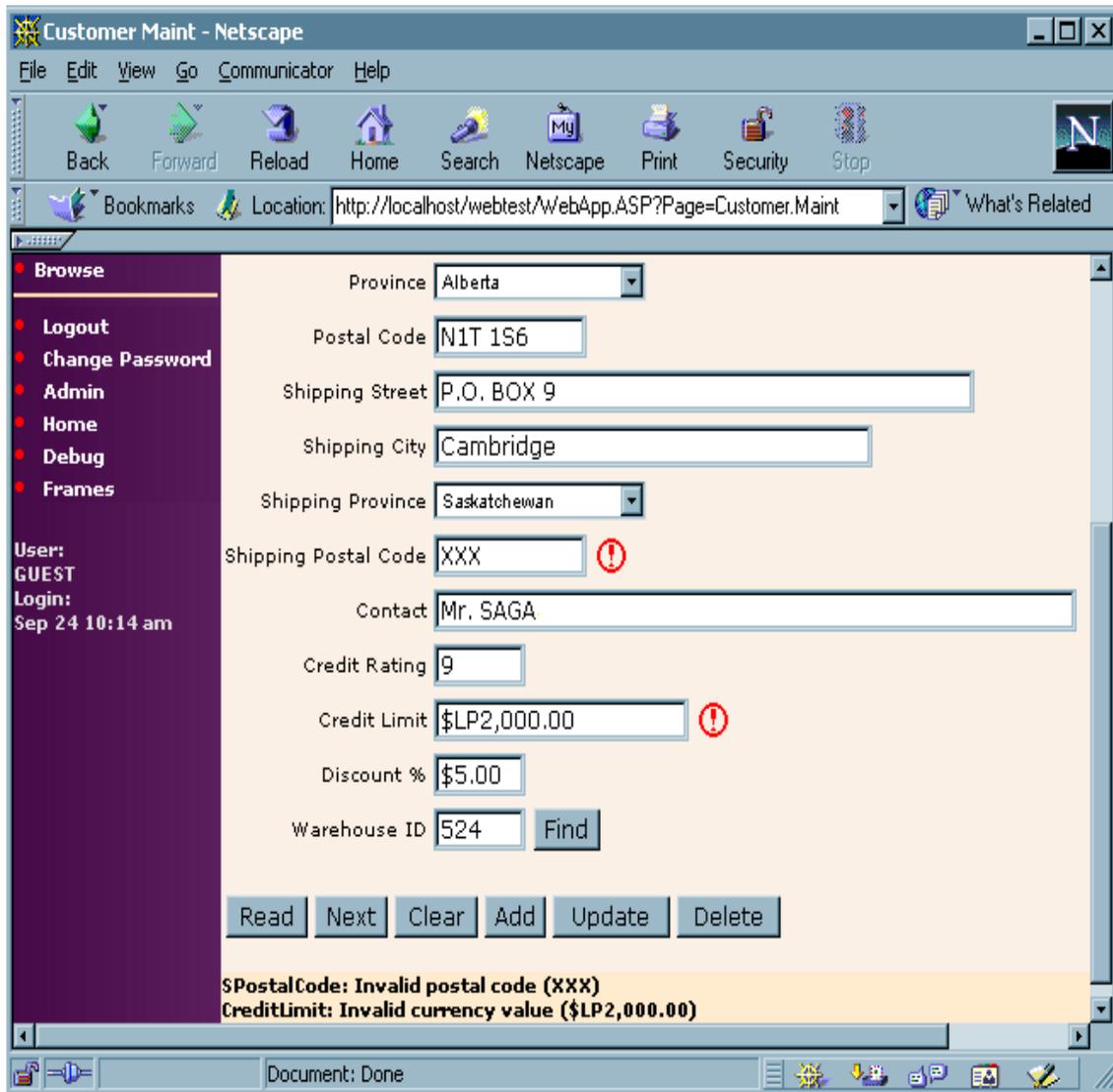
The screenshot displays a web form with two sections: 'Shipping Address' and 'Contacts'. The 'Shipping Address' section includes text boxes for 'Street' (P.O. BOX 9), 'City' (Cambridge), a dropdown for 'Province' (Saskatchewan), and a text box for 'Postal Code' (xxx). The 'Postal Code' text box is highlighted in black. The 'Contacts' section includes radio buttons for 'Credit Rating' (AAA, AA, A, B, C, D, E), a text box for 'Credit Limit' (\$2,P00.00) which is highlighted in black, a text box for 'Discount %' (5), and a text box for 'Warehouse ID' (524) with a 'Find' button. Below the form are buttons for 'Read', 'Next', 'Clear', 'Add', 'Update', and 'Delete'. At the bottom, two error messages are displayed: 'SPostalCode: Invalid postal code (xxx)' and 'CreditLimit: Invalid currency value (\$2,P00.00)'. The error messages are in a dark grey box with white text.

Errors in a Web Application Viewed in Internet Explorer

How Errors are Displayed in Navigator

When an error occurs in a Construct Spectrum web application accessed with Navigator, the pertinent text boxes are followed by a graphic, and an explanation of the error appears in the message area at the bottom of the page. The tool tip for the graphic also displays the error message. To resolve the error, the user must clear the text box or provide correct information.

For example, the following illustration from the Demo web application shows two examples of BDT errors:



Errors in a Web Application Viewed in Navigator

The Debug Page

You can open the Debug page by clicking **Debug** in the navigation bar of your web application. This page provides global and session information that is handy for debugging applications. It also allows you to verify the application's response to validations because it shows cached errors. The following example shows information gathered from the Customer maintenance page in the Demo web application:

The screenshot shows a web browser window titled "Application Debug Data - Microsoft Internet Explorer". The page content is titled "Order Entry Demo" and "Application Debug Data". On the left is a navigation menu with options like "Customer2", "Product", "Warehouse", "Logout", "Change Password", "Admin", "Home", "Debug", and "Frames". The main content area displays a table of application keys and values.

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,P00.00~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

At the bottom left, it shows "User: USER X" and "Login:". At the bottom right, the browser status bar indicates "Local intranet".

Debug Page

Using this example, the Debug page shows the following information that is 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 contained in the Debug page is deleted when you either resolve an error or logout of the application

For information about the other information on this page, see **Debug Page**, page 45.

SECURING YOUR APPLICATION

This chapter describes the security facilities supported by Construct Spectrum and Microsoft Internet Information Server you can use to secure your web application. This chapter also explains how to modify security logic and functionality in Visual Basic using the object factory's user exits and the Globals.bas module.

The following topics are covered:

- **Security Supplied with Construct Spectrum**, page 180
- **Security Supplied by Microsoft Internet Information Server**, page 183

Security Supplied with Construct Spectrum

This section describes the three levels of security supplied with Construct Spectrum.

Spectrum Security

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

For more information, see *Construct Spectrum Administrator's Guide*. For information about users and groups, see **Defining Groups and Users**, page 95, *Construct Spectrum Programmer's Guide*.

Security Sockets

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

Login Functionality

The Login page is supplied with Construct Spectrum web application as a framework component. The Login page uses HTTP security to authenticate the user's 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's 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, but, by modifying the Globals.bas module, you can automatically enable the secure login functionality. For more information about modifying the Globals.bas module, see **Globals.bas**, page 182.

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

Customizing Security

This section describes 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 any of the following:

- 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 the user ID and password, and creates a security profile. The security profile contains a list of Spectrum domains, objects, and methods that the user can access. Add the following code to the user exit to have your application automatically validate a user:

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

IsPermittedCustomTags

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

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

Globals.bas

Add the following line of code to the Globals.bas framework module to enable automatic login functionality:

```
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 **The Key Manager**, page 183.

Security Supplied by Microsoft Internet Information Server

This section provides an overview of the security facilities provided by the web server.

The 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.

The 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. A certificate authority requires you to 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.

The 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 is used to establish a secure SSL connection with a user's web browser, not to encrypt transmitted information.

DEPLOYING YOUR WEB APPLICATION

There are three ways you can deploy your Construct Spectrum web application: manually, using the Package and Deployment Wizard in Visual Basic, or using third party software. This chapter briefly 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 186
- **Manual Deployment**, page 187
- **Using the Package and Deployment Wizard**, page 190

Before Deploying the Application

Before deploying your application, make sure that you have:

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

Manual Deployment

To manually deploy your ABO and web applications, you must perform the following actions:

- 1 Compile the ABO and web projects to create ActiveX and COM dynamic link libraries (DLLs). 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 Register the ABO DLL using Regsrv32. To use this command, select **Run** from the **Start** button. In the **Run** dialog, select **regsrv32** from the drop-down list and enter your application directory.
- 4 Register the web DLL using Microsoft Transaction Server. To register the Spectrum web DLL you must:
 - Create a new package for the web application DLL using the Microsoft Transaction Server. For more information, see **Creating a New Package for the Web DLL**, page 187.
 - Add the web application DLL to the new package and install the application.
- 5 Create a virtual directory using the Microsoft Internet Information Server. You only need to do this if you have not saved your projects under C:\Inetpub\wwwroot. For more information, see **Creating a Virtual Directory**, page 188.
- 6 Create a starting point for the application using the Microsoft Internet Information Server. For more information, see **Creating a Starting Point for the Application**, page 188.
- 7 Modify application settings and properties. This step may not be necessary unless you are web site manager. For more information, see **Modifying Application Settings**, page 188.

Creating a New Package for the Web DLL

- To create a new package for the Spectrum web DLL:
- 1 Open the Microsoft Management Console.
 - 2 Right-click the Microsoft Transaction Server folder, point to **New** and select **Folder** from the shortcut menu.
 - 3 Provide a name for the new package.
This is the package where you will insert the Spectrum DLL.

Creating a Virtual Directory

- To create a virtual directory:
 - 1 Open the Microsoft Management Console.
 - 2 Right-click the Microsoft Internet Information Server folder, point to **New** and select **Virtual Directory** from the shortcut menu.
 - 3 Provide a name for the new folder and a link to your application folder.

Creating 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 and select **Properties** from the shortcut menu. The **<Application Name> Properties** Dialog box appears.
 - 4 Click **Create** to provide a starting point for the application.

Modifying Application Settings

- 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 and select **Properties** from the shortcut menu. Make sure you are on the **Directory** tab.
 - 4 Click **Create**.
 - 5 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 itself, from being affected if this application fails.
 - 6 Click **Configuration** and select the **App Options** tab.

- 7 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 that you can identify the user across pages in the application.

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

- 8 Provide a time-out period. If you specify a high value, the user will not have to log back on again so frequently. However, performance may suffer if the application is heavily used.

Click **Help** to open Microsoft Help if you want more information about how to use this dialog.

Using the Package and Deployment Wizard

You can use Microsoft's 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, you will have to manually add the following application components:

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

Before Using the Package and Deployment Wizard

Before using the wizard, make sure that you have:

- Compiled your ABO project to create a DLL.
- Customized application and ASP file settings. For information about customizing application settings, see **Modifying Application Settings**, page 188.
- 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.

Creating 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
- Make sure the ABO and web DLLs are selected in the **Included Files** dialog box.
- Make sure that the System, Spectrum run-time, and the Visual Basic run-time DLLs are not selected in the **Included Files** dialog box.

Deploying 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, you need to install the Microsoft Posting Acceptor. For more information about installing and configuring the Microsoft Posting Acceptor, read article Q92115 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** dialog box:
 - Graphics folder
 - Support folder
 - HTML files
 - ASP files
- Specify HTTP Post as the publishing protocol.

INDEX

A

ABO

- coding validations in, 172
- role in application architecture, 27

Action bar

- example in Demo web application, 57
- HTML template and page handler, 77

Active Server Page

See ASP

ActiveX Business Object

See ABO

Add as related document option, 117

Add-Ins to Visual Basic, 20

Administration page

- example in Demo web application, 44
- HTML template and page handler, 77

Allow user to keep session data

option, 45

ALTERNATE tag, 150

AppDictionary.bas, 74

customizing, 75

ASP, 73

- role in application architecture, 29
- WebAppF.asp, 73
- WebAppFS.asp, 73

B

BAS files

AppDictionary.bas, 74

customizing, 74

Globals.bas, 74

OFactory.bas, 74

TagProcessing.bas, 74

Utility.bas, 74

BDT

changing name in page handler, 98,
102

sources of information, 13

using in web applications, 172

BestViewed.htm, 77

Browse rows

example in Demo web application, 61

BROWSE tag, 150

BrowseCommon.js, 82

BROWSER tag, 151

BrowseTemplate.htm, 77

C

Calendar pop-up

example in Demo web application, 57

HTML template, 78

Cascading style sheet

StylesIE.css, 76

StylesNav.css, 76

Change Password page

example in Demo web application, 48

HTML template and page handler, 78

CHECKBOX tag, 151

Code

- protecting using implied user exits, 97
- protecting using PRESERVE tag, 97

Common.js, 82

Construct Spectrum

- documentation, 18

Construct Spectrum Administration subsystem

- role in application architecture, 26

Construct Spectrum Programmer's Guide, 13

Construct Spectrum SDK

- documentation, 17
- related products
 - HTML editors, 22
 - Visual Basic Add-Ins, 20
 - web browsers, 22
 - web server management, 23
- using, 20

Construct Spectrum web application architecture of, 24

- Internet/intranet, 30
- mainframe server, 25
- Microsoft Internet Information Server, 27

deploying

- planning for, 32

designing, 33

developing

- development steps, 31
- planning for, 32

example, 37

packaging and deploying, 185

- before deploying, 186
- manually, 187
- using the Package and Deployment Wizard

- deploying the package, 191

planning, 31

role in application architecture, 28

securing, 179

Construct Spectrum web project

- creating, 64
- project directory, 67

Control

- changing, 125
- changing captions, 136
- derivation, 126
- radio button, 130
- section
 - changing views, 131
 - collapsible, 134
- selection list, 127
- text area
 - changing width, 136
- text box
 - changing width, 136

Conventions

- in this guide, 15

CSS

- See Cascading style sheet

Customizing

- BAS files, 74
- HTML before generating, 123
- security in Visual Basic
 - using Globals.bas, 182
 - using the object factory, 181

D

Debug page

- Application key, 47
- example in Demo web application, 45
- HTML template and page handler, 77
- Session key, 47
- viewing cached errors, 177

Demo web application, 37

- accessing, 38

Deploying, 185
 before deploying, 186
 manually, 187
 planning for, 32
 using the Package and Deployment Wizard, 190

DHTML, 22

Dispatch service
 See Spectrum dispatch service

Documentation

 Construct Spectrum, 18
 Construct Spectrum SDK, 17
 Natural Construct, 18

E

Entire Broker
 role in application architecture, 26

ERROR tag, 152

ErrorHandler
 HTML template and page handler, 78

Errors
 showed in web pages, 174

ERRORS tag, 152

Example web application, 37

F

FIELD tag, 153

Find button
 example in Demo web application, 56

Footer
 HTML template and page handler, 78

Frames mode, 50
 Frameset HTML template and page handler, 78

 WebAppF.asp, 73
 WebAppFS.asp, 73

Framework components

 action bar, 57, 77
 Administration page, 44, 77
 ASP components, 73
 BAS files, 74
 Best Viewed page, 77
 Browse template, 77
 calendar pop-up, 57, 78
 cascading style sheets, 76
 Change Password page, 48, 78
 Debug page, 45, 77
 Error Handler page, 78
 Find button, 56
 footer, 78
 Frames, 50
 Frameset, 78
 Global.asa, 73
 header, 54, 78
 Home page, 39, 78
 in Project Explorer, 69
 JavaScript, 82
 KeySelector, 61, 78
 Layout template, 78
 Login page, 42, 79
 Logout page, 79
 Maint HTML template, 79
 message area, 58, 79
 navigation bar, 41, 79
 range options, 61
 section, 54
 WebApp.cls, 73
 WebAppF.asp, 73
 WebAppFS.asp, 73
 WebAppl.cls, 73

Framework HTML templates

 customizing, 107
 description of, 106
 role in application architecture, 29

Framework page handlers

 role in application architecture, 29

G

- Generated code
 - preserving using implied using exits, 97
 - protecting using PRESERVE tag, 97
- Global.asa, 73
- Globals.bas, 74
 - customizing, 75, 182
- GUI keyword, 126

H

- Header
 - example in Demo web application, 54
 - HTML template and page handler, 78
- Home page
 - changing, 170
 - example in Demo web application, 39
 - HTML template and page handler, 78
- HTML editors
 - tested, 22
- HTML Properties dialog
 - Element properties tab, 128
 - Links tab, 139, 142
 - Section Type tab, 135
- HTML tag
 - replacement
 - See also Replacement HTML tag
- HTML template
 - configuring, 114
 - customizing, 107
 - customizing before generation, 123
 - adding links, 136
 - changing captions, 136
 - changing column alignment, 140
 - changing controls, 125
 - changing header text, 143

- changing section views, 131
 - changing width of text boxes and text areas, 136
 - creating links, 140
 - deselecting fields for generation, 125, 140
 - specifying options in selection list, 127
 - specifying radio buttons, 130
 - definition of, 106
 - examples in web page, 80
 - generating, 117
 - in Project Explorer, 121
 - previewing, 120
 - role in application architecture, 29
- HTML Template wizard
 - configuring the template, 114
 - confirming ABO details, 113
 - generating with, 117
 - invoking, 109
 - selecting an ABO, 111
 - using, 109
 - HTTP
 - security, 43
 - HTTPS, 43

I

- IE
 - See Microsoft Internet Explorer
- IEBrowse.js, 82
- IECommon.js, 83
- IEMaint.js, 83
- IIS
 - See Microsoft Internet Information Server

Implied user exit, 97
INDEX tag, 153
INFRAME tag, 154
INSTANCE tag, 154
Internet/Intranet
 role in application architecture, 30

J

JavaScript
 BrowseCommon.js, 82
 Common.js, 82
 IEBrowse.js, 82
 IECommon.js, 83
 IEMaint.js, 83
 NavBrowse.js, 83
 NavCommon.js, 83

K

KeySelector
 and AppDictionary.bas, 75
 example in Demo web application, 61
 HTML template and page handler, 78

L

Layout HTML template, 78
Link
 creating from browse to maintenance
 page, 140
 creating from maintenance field to
 browse page, 136

LINK tag, 76
LOGGEDOUT tag, 155
Login page
 enabling in Page Handler wizard, 94
 example in Demo web application, 42
 HTML template and page handler, 79
LOGIN tag, 154
Logout page
 HTML template, 79
LOOKUP tag, 155

M

Mainframe server
 role in application architecture, 25
Maint HTML template, 79
MAINT tag, 155
Message area
 example in Demo web application, 58
 HTML template and page handler, 79
Microsoft Internet Explorer, 22
 cascading style sheet for, 76
 JavaScript for, 82
 showing errors, 174
 version supported by Construct
 Spectrum web applications, 77
Microsoft Internet Information
Server, 23
 role in application architecture, 27
 security, 183
Microsoft Management Console, 23
 setting session timeout, 43
Microsoft Transaction Server, 23
 role in application architecture, 28
MTS
 See Microsoft Transaction Server

- Multiple edit text area view, 133
- Multiple edit view, 133
- Multiple sort keys
 - example in Demo web application, 61
- Multiple value field, 126

N

- Natural Construct
 - documentation, 18
- Natural subprogram
 - role in application architecture, 25
- NavBrowse.js, 83
- NavCommon.js, 83
- Navigation bar
 - example in Demo web application, 41
 - HTML template and page handler, 79
- Navigator
 - See Netscape Navigator
- Netscape Navigator, 22
 - cascading style sheet for, 76
 - JavaScript for, 83
 - version supported by Construct Spectrum web applications, 77
- Nonframes mode, 50

O

- Object factory
 - OFactory.bas, 74
 - updating using the Object Factory wizard, 165
 - user exits in, 170
 - using to customize security, 181
 - when to update, 164

- Object Factory wizard
 - generating with, 169
 - invoking, 165
 - selecting the page handler, 167
- OFactory.bas, 74

P

- Package and Deployment Wizard
 - before using, 190
 - using to create packages, 190
 - using to deploy the application, 191
- Page handler
 - coding custom browse actions in, 101
 - coding custom maintenance actions in, 100
 - coding validations in, 100, 173
 - customizing, 97, 161
 - framework
 - role in application architecture, 29
 - ID, 94
 - parsing custom tags, 98
 - processing HTML tags, 146
 - role in application architecture, 28
 - role in assembling HTML templates, 106
 - user exits in
 - browse page handlers, 101
 - maintenance page handlers, 97
- Page Handler wizard
 - configuring, 92
 - confirming ABO details, 90
 - generating with, 95
 - invoking, 86
 - selecting an ABO, 88
- PAGE tag, 156
- ParseTemplate.CustomTags, 98
- Periodic group, 126
- Personal Web Server, 23

PRESERVE tag, 97

R

RADIO tag, 156

Range option

example in Demo web application, 61

REPEAT tag, 157

Replacement HTML tag

ALTERNATE, 150

BROWSE, 150

BROWSER, 151

CHECKBOX, 151

creating, 107, 160

ERROR, 152

ERRORS, 152

FIELD, 153

INDEX, 153

INFRAME, 154

INSTANCE, 154

LOGGEDOUT, 155

LOGIN, 154

LOOKUP, 155

MAINT, 155

PAGE, 156

parsing, 98

processing, 102

processing by page handler, 146

RADIO, 156

REPEAT, 157

role in HTML template, 106

SECURITY, 157

SELECT, 158

SUBMIT, 158

syntax of, 147

TITLE, 159

types of, 148

S

Section, 126

collapsible, 134

example in Demo web application, 54

collapsible, 55

view options, 55

multiple edit text area view, 133

multiple edit view, 133

single edit view, 132

single edit with report view, 132

view option

changing, 131

Security

customizing in Visual Basic

using Globals.bas, 182

using the object factory, 170, 181

HTTP, 43

HTTPS, 43

Login page, 42, 94

supplied by IIS, 183

supplied by Microsoft

key manager, 183

server certificate request, 183

SSL Key pair, 183

supplied with Construct

Spectrum, 180

Construct Spectrum Administration
subsystem, 180

login functionality, 180

security socket support, 180

Security service

role in application architecture, 26

SECURITY tag, 157

SELECT tag, 158

Session timeout, 43

Single edit view, 132

Single edit view with report view, 132

Sort key

example in Demo web application, 61

Spectrum Dispatch Client
 role in application architecture, 27
Spectrum dispatch service
 role in application architecture, 26
Spectrum menu, 21
Spectrum web application
 See Construct Spectrum web application
Spectrum web project
 See Construct Spectrum web application
StylesIE.css, 76
StylesNav.css, 76
SUBMIT tag, 158
Subprogram proxy
 role in application architecture, 25
 sources of information, 14

T

TagProcessing.bas, 74
 customizing, 75
 modifying, 160
Text area
 changing width, 136
Text box
 changing width, 136
TITLE tag, 159

U

User exits
 implied, 97
 in browse page handlers, 101
 in maintenance page handlers, 97

Using
 this guide, 13
Utility.bas, 74

V

Validating data, 171
 coding in the page handler, 173
 coding validations in page handler, 100
 in the ABO, 172
 using BDTs, 172
 validation types, 172
Verification rule
 viewing, 130
View button, 131
Visual Basic
 Construct Spectrum SDK Add-Ins, 20
 Spectrum menu, 21

W

Web application
 See Construct Spectrum web application
Web browsers
 differences between, 22
 Microsoft Internet Explorer, 22
 Netscape Navigator, 22
Web server
 supported, 23
WebApp.cls, 73
WebAppF.asp, 73
WebAppFS.asp, 73

Wizards

HTML Template, 109

Object Factory, 165

Page Handler, 86

