Informix Error Messages might also be useful if you do not want to look up your error messages on-line.

On-Line Documentation

Different types of on-line documentation are available:

- Universal Web Connect guides
- On-line help
- Release notes and documentation notes

Universal Web Connect Guides

Universal Web Connect on-line guides introduce Universal Web Connect and explain how to use Universal Web Connect and its associated features to develop applications that dynamically retrieve data from the Informix Dynamic Server database.

On-Line Help

The Data Director for Web on-line help facility provides a detailed, context-sensitive explanation of program features. Use it to explore each menu item. To invoke the on-line help, press F1 whenever you are using any Data Director for Web tool.

Release Notes and Documentation Notes

In addition to the Informix set of manuals, the following on-line Þles supplement the information in this manual.

On-Line File	Purpose
Release notes	Describes any special actions required to install, conÞgure, and use Data Director for Web on your computer. Additionally, this Þle contains information about any known problems and their workarounds.
Documentation notes	Describes features not covered in the manuals or modiÞed since publication.

On-line Þles are located in the **$INFORMIXDIR\ddw** directory.

Related Reading

For additional technical information on database management, see *An Introduction to Database Systems*, by C. J. Date (Addison-Wesley Publishing, 1995).

To learn more about the SQL language, consider the following books:

- *A Guide to the SQL Standard*, by C. J. Date with H. Darwen (Addison-Wesley Publishing, 1993)
- *Understanding the New SQL: A Complete Guide*, by J. Melton and A. Simon (Morgan Kaufmann Publishers, 1993)
- *Using SQL*, by J. Groff and P. Weinberg (Osborne McGraw-Hill, 1990)

To use Data Director for Web, you should be familiar with HTML and Web publishing. If you have limited Web publishing experience, consult an introductory text before you read this manual. The following provide an introduction to HTML and Web publishing:

- *Database Backed Web Sites: The Thinking PersonÕs Guide to Web Publishing*, by Philip Greenspun (Ziff-Davis, 1997)
- *Teach Yourself Web Publishing with HTML in a Week*, by Laura Lemay (Sams Publishing, 1995)

- *The Bare Bones Guide to HTML*, by Kevin Werbach (http://werbach.com/barebones)

Informix Welcomes Your Comments

Please tell us what you like or dislike about our manuals. To help us with future versions of our manuals, we want to know about any corrections or clariÞcations that you would Þnd useful. Please include the following information:

- The name and version of the manual you are using
- Any comments you have about the manual
- Your name, address, and phone number

Write to us at the following address:

Informix Software, Inc.
Technical Publications
300 Lakeside Dr., Suite 2700
Oakland, CA 94612

If you prefer to send electronic mail, our address is:

doc@informix.com

Or, send a facsimile to Technical Publications at:

650-926-6571

We appreciate your feedback.

Chapter

Overview of INFORMIX-Data Director for Web

1

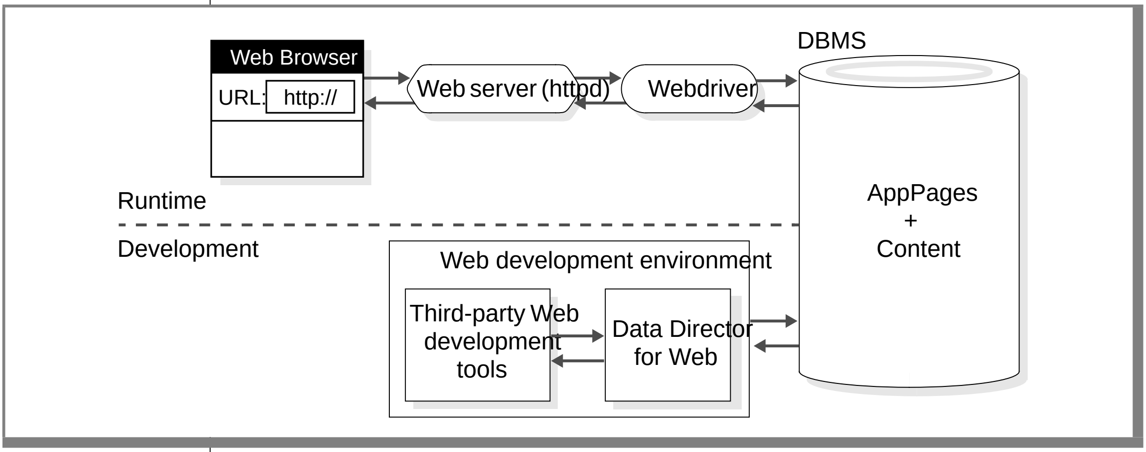
In This Chapter

Data Director for Web is a suite of graphical user interface tools designed to enable you to easily build and manage Informix-based Web sites and applications. This chapter gives an overview of the functionality of the product, the objects it acts on, and the information available in its help system.

Data Director for Web Functionality

Data Director for Web supports the Data Director for Web schema and syntax components of the Web Integration Option. Open interfaces allow you to integrate Web content authoring tools with Data Director for Web, completing your Web development environment. The following diagram illustrates the Informix-based Web development environment.

Figure 1-1
Informix-Based Web Development Environment



Using Data Director for Web, you can:

- edit and preview AppPages and dynamic tags.
- view resources either by Þle groups or by project.
- lock resources and projects, providing developmental concurrency.
- quickly create query and delete tags, option lists, and input and update forms using wizards.
- import Þle-system-based Web sites into an Informix database.
- export database-backed Web sites to the Þle system.
- deploy applications from one database-backed Web site to another.
- integrate third-party Web development tools.

Data Director for Web Objects

Data Director for Web provides a ßexible and extensible base for developing Informix-based Web applications. Data Director for Web acts on objects stored under the Data Director for Web schema. (See Appendix A for detailed schema information.)

Use Data Director for Web to manipulate:

- **Resources.** Web site content, such as AppPages and images, are *resources*. The built-in resources are:
  - **AppPages.** An *AppPage* is an HTML page that can contain Web Integration Option tags, known as *AppPage tags*. By default, an AppPage is stored in the **wbPages** table.
  - **Binaries.** A *binary* is an object like an image or an audio clip. By default, a binary is stored as a large object in the **wbBinaries** table.
  - **Dynamic tags.** A *dynamic tag* is a dynamically expanded AppPage fragment that can be easily shared among multiple AppPages. Dynamic tags are stored in the **wbTags** table.
  - **Other.** This resource describes all objects not stored in the Data Director for Web tables. The resource is stored as a text or binary object in a user table and the source table information is stored in the **wbExtensions** table. A resource of this type cannot belong to a project.
- **Projects.** A *project* is a collection of related resourcesÑsuch as AppPages, binaries, and dynamic tagsÑassociated with a particular Web application. A resource can belong to one or more projects, but is not required to belong to any project. The information speciÞc to a given project is stored in the **wbProjects** table.
- **Preview conÞgurations.** A *preview conÞguration* contains the client-side variable conÞguration settings to enable you to display AppPages in a Web browser from AppPage Editor. Preview conÞguration information is stored in the **wbPreviews** table.
- **Extensions.** An *extension* speciÞes mapping information for resources. You add a new MIME type by deÞning an extension for it, including the name of the table in which the objects reside. Generally, objects should reside in the **wbPages** and **wbBinaries** tables, but existing database tables with resource content can also be used. Tables other than **wbPages** and **wbBinaries** can be associated with only a single extension. Extensions are stored in the **wbExtensions** table.

- **AppPage tags.** An *AppPage tag* is an SGML-compliant tag that enables SQL statements to be executed dynamically within AppPages. AppPage tags are built into the Web Integration Option and therefore do not need to be stored in the Data Director for Web schema.
- **Variable processing functions.** A *variable processing function* is used in a variable expression to enable calculations to be performed using variables passed into an AppPage, generated within an AppPage, or returned from the database. Variable processing functions are built into the Web Integration Option and therefore do not need to be stored in the Data Director for Web schema.

Data Director for Web On-Line Help

This manual shows you how to get started using Data Director for Web. It provides an introduction to the two core components, Site Manager and AppPage Editor. It also provides examples of common tasks you might want to perform using Data Director for Web. It does not, however, provide detailed reference material about Data Director for Web menus and toolbars.

For detailed reference material, use the on-line help provided with Site Manager and AppPage Editor. You can access help by:

- pressing F1.
- choosing the **Help** menu option.

Chapter

Setting Up Your Data Director for Web Environment

2

In This Chapter

Data Director for Web is a suite of client tools designed to enable you to easily build and manage Web applications stored in an Informix database. Before you run any of the Data Director for Web tools, you must conÞgure your client and server installations as discussed in this chapter.

Before You Install Data Director for Web

Prior to installing Data Director for Web, verify that you have working installations of:

- Informix Dynamic Server (with or without Universal Data Option)
- Web Integration Option (Informix Web DataBlade module or INFORMIX-Universal Web Connect)
- INFORMIX-Connect client libraries and the INFORMIX-DBA kit

UWC

You must conÞgure your Informix Dynamic Server (without Universal Data Option) installation with a blobspace. When installing the Data Director for Web schema, you must know the name of this blobspace. ♦

WB

You must conÞgure your Informix Dynamic Server with Universal Data Option installation with an sbspace. When installing the Data Director for Web schema, you must know the name of this sbspace. ♦

See the Data Director for Web release notes for version compatibility information.

ConÞguring Data Director for Web

To conÞgure Data Director for Web, you must modify the conÞguration information on both your client machine and database server (Web server) machine.

Updating the Web Integration Option Server ConÞguration

To enable Web content to be served from the Data Director for Web schema using the Web Integration Option, add the conÞguration variables described in this section to the conÞguration Þle on your server machine.

UWC

Add the variables described in this section to the **wcconÞg.std** Þle. For more information on this conÞguration Þle, see the Universal Web Connect on-line documentation. ♦

WB

Add the variables described in this section to the **web.cnf** Þle. For more information on this conÞguration Þle, see the *Informix Web DataBlade Module UserÕs Guide*. ♦

Setting MI_WEBTAGSSQL

Set this variable to deÞne the table in which the dynamic tags are stored.

To deÞne the **wbTags** table as the default dynamic tag storage, add the following line to the server conÞguration Þle:

```
MI_WEBTAGSSQL SELECT parameters, object FROM wbTags WHERE
    webupper(ID)=webupper(Õ$MI_WEBTAGSIDÕ);
```

WB

Setting MI_WEBSCHEMADEF

Set this variable to deÞne the schema in which content is stored: the Data Director for Web schema or AppPage Builder (APB) schema.

To deÞne the Data Director for Web schema as the schema in which content is stored, add the following line to the server conÞguration Þle:

```
MI_WEBSCHEMADEF wb
```

Setting MI_WEBEXTENSIONS

Set this variable to enable or disable caching of extension information from the **wbExtensions** table.

Important: For optimal performance, you should disable caching only in a development environment where you modify the extensions frequently.

To enable caching of extension information, add the following line to the server conÞguration Þle:

```
MI_WEBEXTENSIONS /full/path/to/cache/file.ext
```

Replace ***/full/path/to/cache/Þle.ext*** with the path appropriate to your installation.

To disable caching of extension information, add the following line to the server conÞguration Þle:

```
MI_WEBEXTENSIONS nocache ♦
```

Setting Additional Server ConÞguration Variables

UWC

If you are using Universal Web Connect, you must perform the following additional steps.

To conÞgure Data Director for Web with Universal Web Connect

1. Stop the Universal Web Connect service.
2. Copy **$INFORMIXDIR\BIN\wcdaemon.exe** to **wbdaemon.exe**.

3. Update your **wcconÞg.std** Þle to include the following application section:

```
#application section
#applies to any Webdriver
APP wbdaemon.exe 1
MI_WEBDRIVER      Y
MI_WEBBINDIR      $INFORMIXDIR\bin
MI_WEBWORKDIR     $INFORMIXDIR\bin
MI_SERVER         db_name@ol_server # database@server
MI_USER           informix
MI_PASSWORD       informix
MI_WEBSCHEMADEF   wb
MItab             wbPages
MIcol             object
MInam             ID
MIval             ddw
MItabObj          wbBinaries
MIcolObj          object
MInamObj          ID
MItypeObj         image/gif
ENDAPP
```

4. Restart the Universal Web Connect service. ♦

WB

If you are using the Web DataBlade module and you have AppPages stored in the AppPage Builder schema, create a new conÞguration Þle in a separate Webdriver directory on your server for your Data Director for Web schema.

You must access each schema through its own conÞguration Þle. Add the variables described in the preceding pages of this section to the new conÞguration Þle. For more information on AppPage Builder and the AppPage Builder schema, see the *Informix Web DataBlade Module UserÕs Guide*. ♦

Setting Additional Variables for the NSAPI Webdriver

WB

If you use the NSAPI Webdriver, you must set MI_WEBNSNAMETRANS and WEB_HOME based on the value of the **nametrans** directive in the Netscape **obj.conf** Þle.

WEB_HOME is set to a URL-mapped path that is a subdirectory of the path speciÞed in the **from** parameter of the **nametrans** directive. This URL-mapped path forms the root of the URL used to access your Web site; the actual path need not exist on the Þle system. MI_WEBNSNAMETRANS is set to the value of WEB_HOME. For example, given the following entry in the Netscape **obj.conf** Þle:

```
Nametrans from="/root" fn="pfx2dir"
dir="/home/web" name="Webdriver"
```

WEB_HOME and MI_WEBNSNAMETRANS might be set as follows:

```
WEB_HOME /root/mydir
MI_WEBNSNAMETRANS /root/mydir
```

And the URL for the AppPage `/my/project/myapp.html` in this installation would be:

```
http://root/mydir/my/project/myapp.html
```

For more information on the NSAPI Webdriver, see the *Informix Web DataBlade Module UserÕs Guide*. ♦

Setting Up the Preview ConÞguration on the Client

The Data Director for Web client requires a subset of the information about the corresponding server conÞguration to enable AppPage Editor Preview functionality. This conÞguration information is stored in the **wbPreviews** table. When you install Data Director for Web, a template preview conÞguration named **Default** is created.

To enable the AppPage Editor Preview functionality, you must edit the template preview conÞguration settings from the **Tools→Preview Settings** menu in AppPage Editor or Site Manager to match the settings for the server that contains the resources associated with your project.

You can create additional preview conÞgurations to associate projects with your server installations, and you can change the preview conÞguration for a project within the **Preview Settings** tab of the Edit Project dialog box.

UWC

Set the MI_WEBTAGSSQL variable in the preview conÞguration to match the value in the **wcconÞg.std** server conÞguration Þle. Set the UWC_HOME variable to the fully qualiÞed path to the Universal Web Connect service, including the host name. An example of a valid UWC_HOME setting is:

```
http://localhost/ifmx-bin/wcdrvr.exe/wbdaemon.exe
```

For more information on deÞning UWC_HOME, see the Universal Web Connect on-line documentation. ♦

WB

Set the MI_WEBTAGSSQL variable in the preview conÞguration to match the value in the **web.cnf** server conÞguration Þle. Set the WEB_HOME variable to the fully qualiÞed path to Webdriver, including the host name. An example of a valid WEB_HOME setting is:

```
http://webserver/cgi-bin/<database>/webdriver
```

For more information on deÞning WEB_HOME, see the *Informix Web DataBlade Module UserÕs Guide*.

If you are using the NSAPI Webdriver, set the MI_WEBNSNAMETRANS variable to match the value of the **nametrans** directive in your server conÞguration Þle. ♦

Chapter

Using Site Manager

3

In This Chapter

The Data Director for Web Site Manager helps you develop and manage Informix-based Web sites and create new Data Director for Web resources. You view your Web site using a Þle system paradigm, similar to that of Microsoft Windows Explorer, or a project paradigm. Read this chapter to learn what you can do with Site Manager and how to do it.

Site Manager Main Window

The Site Manager main window consists of two frames. The left frame provides a tree view of the network environment; the right frame provides a resource view. When fully expanded, these views provide you with a list of all available database servers, databases, and database resources.

The left frame is tabbed, allowing you to view your databases as a group of projects (**Project View**) or as a Þle system (**File View**). The right frame is populated with the objects found at the level selected in the left frame. The resource view consists of the properties for the given set of resources.

Project View

In the **Project View**, you view a database as a group of projects. Each project contains folders for each of the different Data Director for Web supertypes, such as image or text. When you install the Data Director for Web schema into a database, two default folders are created: a **Not In Projects** folder and an **Other Resources** folder. The **Not In Projects** folder contains all resources not associated with any project. The **Other Resources** folder contains all resources stored in the database and mapped in the **wbExtensions** table, but not stored in the Data Director for Web schema.

File View

In the **File View**, you view databases as individual Þle systems. Expanding a folder displays the next level of subfolders.

Menus and Toolbars

The Site Manager menu bar contains the following menus.

Menu	Commands
File	Create a new resource, project, or dynamic tag
Edit	Perform clipboard operations; lock / unlock a resource; edit a resource
View	Invoke external browser for resource preview; conÞgure resource view
Tools	Export from / import to Þle system; migrate AppPage Builder resources to Data Director for Web; conÞgure preview settings
Help	Use the help system

See the appropriate on-line help topics for details concerning the menu options.

Additionally, there are two ÒßoatingÓ context menus that are displayed when you click the right mouse button; a different context menu is displayed depending on the frame in which you click. These menus provide quick access to common functions including clipboard operations, resource locking, and resource preview.

Site Manager has two toolbars that are shortcuts to many menu options, such as resource locking, resource editing, or resource creation. See the appropriate on-line help topics for details.

You can move any toolbar independently of other elements in the user interface. You can also attach it to the frame of the main window. To move a toolbar, drag it to the new location.

Tip: To move a toolbar, you must grab the toolbar itself, not a button on the toolbar.

Connecting to a Server

You can connect to multiple database servers and databases through a single instance of Site Manager. While you provide login information at the database server level, connections are maintained at the database level, one per database. You must have login access on the host machine. (See the *Informix Client Products Installation Guide for Microsoft Windows Environments* for details on accessing your database server).

Database connections are maintained until you exit Site Manager (or the network connection is lost.) To connect to a database server, click the appropriate server icon in the network view.

Installing the Data Director for Web Schema

After you install the Data Director for Web schema in a database using Site Manager, you can connect to the database. When the schema installation window appears, click **OK** to install the Data Director for Web schema into the database.

Important: You must have already installed the Web Integration Option (Universal Web Connect or Web DataBlade module) prior to installing Data Director for Web.

Locking a Project or Resource

You lock a project or resource to prevent other users from updating or deleting resources while you are editing them. When you lock a project, you lock all resources within the project.

You might not be able to obtain a project lock if one or more of the resources in the project is already locked by another user. This can occur when the other user has locked resources individually, or when a project lock is taken on a project that contains resources also in the project you wish to lock. (See ÒData Director for Web ObjectsÓ on page 1-4 for more information on the relationship between projects and resources.)

Data Director for Web provides a simple versioning scheme. A resource stored in the Data Director for Web schema has a version number. This version number increases each time you modify the resource.

Importing a Project

You can import an entire project into the database from the Þle system using the **Import** menu option. On import, the Þle system-based links can all be converted to Webdriver-compatible links valid in the database through a process called *link conversion*. The URL is changed to a Webdriver URL, using a full relative path.

Exporting a Project

You can export a project from the database to the Þle system. All content stored in the Data Director for Web schema and mapped in the **wbExtensions** table is exported.

On export, the dynamic links can all be converted to static links valid on the Þle system through a process called *link conversion*.

Adding a Resource to a Project

You add a resource to a project by dropping the resource icon onto the appropriate project. You can drag a resource from Microsoft Windows Explorer or Site Manager. A resource can belong to more than one project. Dragging an icon from one project to another creates another reference to the resource.

Deleting a Resource

When you delete a resource, you can delete it from a project, or from the database entirely. To delete a resource from the database, you delete it using the **Edit→Delete** menu option. To remove a resource from a project, delete it using the **Edit→Remove from Project** menu option.

Integrating Tools

You extend Site Manager by specifying third-party applications to be used for speciÞc types of content. A new or edit resource operation invokes the appropriate third-party tool based on the extension of the resource.

To map a new resource extension, use the **Extensions** menu option. You need to supply the extension, a name, a fully qualiÞed MIME type, a helper application, the name of the table in which the data is stored, the name of the column in which the actual resource is stored, and the retrieval method for the resource.

Migrating from AppPage Builder

AppPage Builder, an Informix-based Web application that enables you to create and maintain the AppPages that make up your Web projects, is included with the Web Integration Option. AppPage Builder provides a limited subset of Data Director for Web functionality and uses its own schema to store Web content. To migrate existing AppPage Builder applications to the Data Director for Web schema, use the **Migrate from APB2** menu option.

AppPage Builder does not associate dynamic tags with projects. When you Þrst migrate an existing AppPage Builder project into your Data Director for Web schema, all dynamic tags stored in the database are copied and associated with the target project. If you subsequently migrate additional AppPage Builder projects in the same database, the dynamic tags are not copied again; you need to manually copy them into the Data Director for Web schema.

Chapter

Using AppPage Editor

4

In This Chapter

The Data Director for Web AppPage Editor helps you to easily create and edit AppPages and dynamic tags, and to add AppPage tags to existing HTML pages. This chapter explains how to conÞgure and use AppPage Editor.

AppPage Editor Window

AppPage Editor has a single, main window where you create, edit, and view AppPages and dynamic tags. You view the current open document within this window.

Menus and Toolbar

The AppPage Editor menu bar contains the following menus.

Menu	Commands
File	Connect to a database; open, save, and print AppPages
Edit	Copy and Þnd text
Insert	Insert a tag, resource, or link
Tools	Run **WebLint**, **WebExplode**, or an external browser; invoke wizards; set preview settings; set user preferences
Navigate	Use browser controls
View	Show or hide the toolbars and status bar; toggle **Edit** and **Preview** modes
Help	Use the help system

See the appropriate on-line help topics for details concerning the menu commands.

Additionally, there is a “floating” context menu that is displayed when you click the right mouse button with the cursor in the main window. This menu provides quick access to common functions including clipboard operations, dynamic tag editing, and user preferences.

AppPage Editor has four toolbars that provide shortcuts to many menu commands, such as connecting to a database, navigating with the browser, or invoking external applications. See the appropriate on-line help topics for details on the following toolbars:

- Standard toolbar
- Tag toolbar
- Search toolbar
- Tools toolbar

You can move any toolbar independently of other elements in the user interface. You can also attach it to the frame of the main window. To move a toolbar, drag it to the new location.

Tip: To move a toolbar, you must grab the toolbar itself, not a button on the toolbar.

Configuration Options

Configure AppPage Editor settings with the **Tools→Preferences** menu. These preferences include:

- the method of color-coding HTML text and special tags.
- the colors used in color-coding AppPage tags and HTML text.
- whether to enable or disable META tag information.
- whether to enable or disable **WebExplode** in AppPage preview.

Once set, these preferences are persistent across all subsequent invocations of AppPage Editor.

Use the **Tools→Preview Settings** menu to configure the database and project settings to use for AppPage preview.

Edit and Preview Views

You can toggle the main window between two views: **Edit** and **Preview**.

Edit

The **Edit** view displays the source text of the AppPage or dynamic tag you are currently editing. AppPage Editor features are focused on this view. Within this view, you can:

- create a new AppPage or dynamic tag.
- edit an existing AppPage or dynamic tag.
- perform a search and replace against an AppPage or dynamic tag.
- use wizards to quickly create new query and delete tags, input and update forms, and option lists.
- insert AppPage tags into an AppPage or dynamic tag.
- insert other Data Director for Web resources into an AppPage or dynamic tag.
- add variable processing functions to an AppPage or dynamic tag.
- run **WebLint** to check the syntax of an AppPage or dynamic tag.
- print an AppPage.

In the **Edit** view, you can conÞgure AppPage Editor so that HTML text is color-coded to highlight AppPage tags.

Preview

The **Preview** view displays a Web browser that contains the rendered HTML of the current AppPage. By default, you view the AppPage expanded by **WebExplode**. You can also choose to view the unexpanded HTML source.

For **Preview** to operate correctly with **WebExplode**, you must have the appropriate client-side variable settings, described in Chapter 2, ÒSetting Up Your Data Director for Web Environment,Ó in the preview conÞguration. You can edit or import the preview conÞgurations from the **Tools→Preview Settings** menu.

You navigate in the browser from either the menu or the toolbar.

AppPage Editor Features

You can perform the following tasks using AppPage Editor:

- Connecting to a database
- Creating an AppPage
- Creating a dynamic tag
- Using the wizards
- Editing a wizard-generated dynamic tag
- Editing an existing AppPage or dynamic tag
- Previewing in an external browser
- Running **WebLint** to check for syntax errors
- Converting Þle system links

Each task is described in the following sections.

Connecting to a Database

You can run AppPage Editor as a stand-alone application or as a database client. In stand-alone mode, AppPage Editor acts as a text editor for Þles on the local Þle system only.

Important: You must install the Data Director for Web schema using Site Manager prior to connecting to a database with AppPage Editor.

Each instance of the AppPage Editor supports a single database connection with a single AppPage or dynamic tag open at a time. When you open or create a different resource, you must Þrst close the current resource. To edit multiple resources concurrently, invoke multiple instances of AppPage Editor.

You connect to a database server from AppPage Editor in one of two ways:

- Supply the database server name, the database name, and your user name and password for the host on which the database server resides. (See *Informix Client Products Installation Guide for Microsoft Windows Environments* for details on setting database server information).

- Choose a connection from the most recent database accesses list on the **File** menu.

Creating an AppPage

The main function of AppPage Editor is to help you build AppPages quickly and easily. You create the AppPage, add resources, and save the AppPage to either the database or the Þle system.

AppPage Properties

You deÞne attributes of an AppPage through its properties. The attributes you can deÞne are:

- **Description.** A text description of the AppPage.
- **Author.** The name of the author of the AppPage.
- **Version.** The version number of the AppPage.
- **Keywords.** Keywords describing the AppPage.
- **Read Level.** The authorization level required by an end user to view the AppPage in a Web browser.

In addition, there are attributes you cannot set directly, that are set automatically by the Data Director for Web tools. These attributes are:

- **Last Changed.** The date and time that the AppPage was last modiÞed.
- **Changed By.** The name of the user who last modiÞed the AppPage.
- **Last Locked.** The date and time that the AppPage was last locked.
- **Locked By.** The name of the user who currently has the AppPage locked.

META tag information

You can embed information about an AppPage into the AppPage using META tags. AppPage Editor embeds this meta-information into the AppPages that it saves.

META tag	Information
Description	Description of the AppPage
Keywords	Keywords associated with the AppPage
Author	Author of the AppPage
Version	Version number of the AppPage used to create current AppPage

The Description and Keywords META tags are standard HTML META tags used by Web crawlers to extract information about the page. The Author and Version META tags are used only by Data Director for Web tools.

You choose whether or not to store META tags in AppPages through the User Preferences dialog box.

Creating a Dynamic Tag

You create dynamic tags either manually or automatically. When creating a dynamic tag manually, you use AppPage Editor the same way that you would when creating a new AppPage.

In addition, you can use the following AppPage Editor wizards to automatically generate dynamic tags to perform database operations:

- Query
- Delete
- Input
- Update
- Option List

Using the Wizards

AppPage Editor offers wizards to help you quickly build query tags, delete tags, input forms, update forms, and option lists. Each wizard allows you to visually create the tag or form, and to choose an output format. The wizard dynamically builds the HTML to perform the appropriate actions; you choose to either store the resultant text in the database as a dynamic tag or insert it directly into the AppPage.

For examples of creating tags using the wizards, see Chapter 5, ÒBuilding an Application with Data Director for Web.Ó

Query

The Query wizard guides you through creating an MISQL tag to query against underlying tables. You provide the tag name, output format, table names, column names to be retrieved, column names to be displayed, selection criteria, and any necessary hyperlinks.

Delete

The Delete wizard guides you through creating an MISQL tag to delete information from an underlying table. You provide the tag name, table name, selection criteria, and success message.

Input

The Input wizard guides you through creating HTML forms to insert new rows into underlying database tables. You provide the table name, column names, and success message, and you choose the output format.

The Input wizard generates two distinct tags. The Þrst (*tagname*_insert) creates the user input form; the second (*tagname*_add) validates the user input and inserts the data into the database. Execution of the tags is controlled through the **apb_state_** variable. The code for *tagname*_insert is contained within an MIBLOCK that is executed when **apb_state_** is set to GET; the code for *tagname*_add is contained within an MIBLOCK tag that is executed when **apb_state_** is set to POST. *tagname*_add should always precede *tagname*_insert, either in an AppPage or in the order of invocation.

Update

The Update wizard guides you through creating HTML forms to update information in underlying database tables. You provide the tag name, table name, column names, and success message, and you choose the output format.

Like the Input wizard, the Update wizard generates two distinct tags. The Þrst (*tagname*_update) creates the user input form and populates it with the current information; the second (*tagname*_edit) validates the user input and updates the data in the database. Execution of the tags is controlled through the **apb_state_** variable. The code for *tagname*_update is contained within an MIBLOCK that is executed when **apb_state_** is set to GET; the code for *tagname*_edit is contained within an MIBLOCK tag that is executed when **apb_state_** is set to POST. *tagname*_update should always precede *tagname*_edit either in an AppPage or in the order of invocation.

Option List

The Option List wizard guides you through creating a check box list, radio button list, or selection list. You provide the list name, type of list, SQL query, selection criteria, and formatting information.

Editing a Wizard-Generated Dynamic Tag

Edit dynamic tags created with the Input Form, Update Form, Query, Delete, or Option List wizards using the AppPage Editor Wizard Properties dialog box.

To modify the wizard properties of a dynamic tag while you are editing the dynamic tag in AppPage Editor, click the **Edit→Wizard Properties** menu option. To modify the wizard properties of a dynamic tag embedded within an AppPage, select the dynamic tag and click the **Edit→Wizard Properties** menu option.

Choosing the **Edit→Wizard Properties** menu option opens a tabbed dialog box speciÞc to the wizard used to generate the dynamic tag. Each tabbed page corresponds to one of the windows in the wizard dynamic tag generation sequence. Once you have made your changes, click **OK**. You must save the dynamic tag back to the database to save the changes.

Warning: If you have made manual changes to the dynamic tag, they are overwritten when you regenerate the tag using the wizard.

Editing an Existing AppPage or Dynamic Tag

You edit an existing AppPage or dynamic tag by opening the resource with AppPage Editor.

Clipboard Operations

You can perform standard Windows clipboard operations (cut, copy, and paste text) using AppPage Editor.

Find and Replace

You can perform standard Windows search operations using AppPage Editor. There are two options: **Find**, and **Find and Replace**. Using **Find and Replace**, you search for an occurrence of a text string and then have the option of selectively replacing or replacing all occurrences.

Editing Properties

You can edit the properties of both AppPages and dynamic tags using AppPage Editor. To edit the properties of a dynamic tag, you must Þrst open the tag, instead of simply selecting the tag from an AppPage.

Matching MIBLOCK Tags

You move between <?MIBLOCK> <?/MIBLOCK> pairs in AppPage editor using the **Edit→Match MiBlock** menu option or the shortcut CTRL-B. To Þnd the matching <?/MIBLOCK> tag, click or select the opening <?MIBLOCK> tag. Choose the menu option or shortcut, and the matching <?/MIBLOCK> tag is highlighted. If the tag is unmatched, an error is returned.

Automatic Indentation Within MIBLOCK Tags

Using the **Tools→Auto Indent** menu option, you can indent your MIBLOCK code for readability. When you choose Auto Indent, all MIBLOCK code on the AppPage is indented. Code in nested MIBLOCK statements are indented appropriately.

Previewing in an External Browser

In addition to the **Preview** view, you can invoke an external browser from AppPage Editor to view an AppPage. AppPage Editor automatically locates and invokes your default browser.

Running WebLint

You can run the **WebLint** function to check for syntax errors in AppPage tags. The results are displayed in a read-only, ÒßoatingÓ window. Line numbers are displayed where errors are found, to allow you to easily edit the incorrect syntax. Double-clicking an error message in the ÒßoatingÓ window sets focus to the edit window on the line with the incorrect syntax.

Tip: *You can attempt to execute an AppPage even if **WebLint** reports errors in the AppPage.*

Converting File System Links

Data Director for Web supports both standard Þle system URLs and Webdriver-compatible URLs. A Webdriver-compatible URL is a URL of the form:

UWC

```
http://webserver/ifmx-
bin/wcdrvr.exe/wbdaemon.exe?MIval=$WEB_HOME/path/filename.ext ♦
```

WB

```
http://webserver/cgi-bin/webdriver?MIval=$WEB_HOME/path/filename.ext ♦
```

When you import a project from the Þle system into the database, you convert the links embedded in AppPages using the **Tools→Convert Links in Page** menu option.

Chapter

5

Building an Application with Data Director for Web

In This Chapter

This chapter describes how to use Data Director for Web to develop a simple home page application.

The goal is to demonstrate the functionality of Data Director for Web, rather than to provide a comprehensive SQL or Web Integration Option tutorial.

For more information on SQL, refer to *Informix Guide to SQL: Tutorial*. For more information on the Web Integration Option, refer to the *Informix Web DataBlade Module User's Guide* or the Universal Web Connect on-line documentation.

Before You Begin

You can Þnd the examples described in this chapter in the following directories:

UWC

$INFORMIXDIR\ddw\Examples\Non-UDO\gary ♦

WB

$INFORMIXDIR\ddw\Examples\UDO\gary ♦

For the rest of the chapter, directory references are relative to the preceding directories.

You must create the underlying tables for the example application data before you begin developing the application. The SQL setup Þles that create these tables are found in the **sql** directory.

Important: Run all of the SQL setup Þles in SQL Editor prior to attempting this tutorial.

The images for the example application are found in the **images** directory.

Sample AppPages (HTML Þles) for this example application are located in the **groceries**, **home**, **locations**, and **menus** directories. When a portion of the application requires extensive typing, you might want to simply cut-and-paste from these HTML Þles. Additionally, the complete source code for the example application is listed in Appendix C of this manual.

Scenario

Imagine that Chef Gary has hired you to take his gourmet health food business Òhigh techÓ by putting an internal home page on the World Wide Web. His data is stored in an Informix Dynamic Server database with the Web Integration Option installed. He wants his employees to use the Web site to:

- give telephone customers the location of the store nearest them.
- update grocery inventory.

Building the Foundation

Before you build the application, you must create its basic building blocks, including:

- a project for your resources.
- binary resources, such as images.
- dynamic tags, to provide consistency.

As you proceed through the examples, you might notice that the same result is achieved using several different mechanisms. This is to demonstrate the different ways to accomplish similar tasks using Data Director for Web.

Creating a Project

This section describes how to create a new project in Site Manager.

To create a new project called Gary

1. Choose **File→New→Project**.

2. Type a project name (`Gary`) and description (`GaryÕs Gourmet Home Page`) in the dialog box.
3. Click **OK**.

You have created a new project called **Gary**.

*Tip: To view or modify the preview conÞguration settings for a project, click the **Preview Settings** tab. The **Default** preview conÞguration is selected by default.*

Creating Binary Resources

You have two images for your application: the corporate logo (carrots) and a map of the United States. These images are located in the **images** example directory. To insert these images into the database, you create new binary resources in Site Manager. There are two ways to create a new binary resource:

- Use the **File→New→Resource** menu option.
- Drag the resource from the Microsoft Windows Explorer.

To add the map of the United States, using the Þrst method

1. Click the project **Gary** icon to designate it as the project for the new resource.
2. Choose **File→New→Resource**.
3. Type the resource name (`usa`), pathname (`/gary/images`), and extension (`gif`).
4. Click the **Properties** tab and type the description (`map of the United States`).
5. Use **Browse** to select the Þle **usa.gif** from the Þle system.
6. Click **OK**.

You have created an image resource **usa** with the path **/gary/images** in the project **Gary**.

*Important: You must highlight the project **Gary,** or the **usa** resource is added to **Not in Projects**.*

To add the corporate logo, using the second method

1. Drag **carrots.gif** from the Microsoft Windows Explorer window to the Site Manager window.
2. Drop **carrots.gif** on the project **Gary**.

You have created an image resource **carrots** with the default path (/) and the extension **gif**. Only resource name, pathname, and extension are required attributes for a binary resource.

To set the optional attributes of the carrots resource

1. Expand the **Gary** project to see the MIME type folder list.
2. Expand the **Image** folder to see the image list.
3. Click the Þle icon for **carrots** to highlight it.
4. Choose **Edit→Properties**.
5. Type the attribute information in the appropriate Þelds within the dialog box. For example, set **Description** to `corporate logo`.
6. Click **OK**.

Change to the File view and move **carrots** to the **/gary/images** folder by dragging the **carrots** resource and dropping it into the **/gary/images** folder.

You have created the two image resources needed for the application.

Creating and Editing User Dynamic Tags

You create three user dynamic tags for the application: a dynamic tag to display a standard header, a dynamic tag to display a standard footer, and a dynamic tag to display an image map.

You can create new dynamic tags in both Site Manager and AppPage Editor. Use Site Manager to create simple user dynamic tags, such as user dynamic tags containing only standard HTML. Use AppPage Editor features to create more complex user dynamic tags.

Creating the Header User Dynamic Tag

First, use Site Manager to create the **header** user dynamic tag. This user dynamic tag contains the HTML to set the title for each page.

To create the header user dynamic tag

1. Click the project **Gary.**
2. Choose **File→New→Dynamic Tag.**
3. Type the ID (`header`), description (`header dynamic tag for GaryÕs Gourmet Home Page`), and parameter (`title`).
4. Click the **Body** tab.
5. Type the text for the dynamic tag:

```
<HTML>
<HEAD>
<TITLE>@title@</TITLE>
</HEAD>
<BODY>
<CENTER><EM>Low-fat is where it's at!</EM></CENTER>
<HR>
<P>
```

6. Click **OK**.

You have created the **header** dynamic tag. For information on passing parameters to user dynamic tags, see the *Informix Web DataBlade Module UserÕs Guide* or Universal Web Connect on-line product documentation.

Creating the Footer User Dynamic Tag

Next, create the **footer** dynamic tag. The **footer** dynamic tag contains links to other Web sites. You can use Site Manager to create an empty dynamic tag and enter the tag body using AppPage Editor.

To create an empty footer user dynamic tag

1. Click the project **Gary.**
2. Choose **File→New→Dynamic Tag.**
3. Type the ID (`footer`) and description (`footer dynamic tag for GaryÕs Gourmet Home Page`).
4. Click **OK**.

To edit the dynamic tag

1. Click the project **Gary.**

2. Click the folder **Dynamic Tags**.
3. Double-click the **footer** dynamic tag to open it in AppPage Editor.
4. Type the source code for the dynamic tag:

```
<P>
<HR>
<P>
<CENTER>Many thanks to our sponsors:<BR>

Harry's House of Hummus |
<A HREF="http://www.informix.com">Informix Corp.</A>
<P>
</CENTER>

</BODY>
</HTML>
```

5. Choose **File→Save**.

The default action for **Save** is to save the AppPage or dynamic tag back to its source. In other words, if the AppPage was opened from the database, it is saved back to the database; if the AppPage was opened from the Þle system, it is saved back to the Þle system.

Creating the Image Map User Dynamic Tag

Finally, create the **imgmap** user dynamic tag. This user dynamic tag displays an image map, or an image that, when clicked, returns the coordinates of the click. Create this user dynamic tag in AppPage Editor, using the built-in Data Director for Web dynamic tag **wbImg** as a base.

To retrieve the wbImg tag

1. Click the **Open from DB** button on the toolbar.
2. Select **No Project** from the **Project** drop-down list.
3. Select **Dynamic Tag** from the **Resource** drop-down list.
4. Click **wbImg.tag**.
5. Click **OK**.

To copy the MISQL tag to the clipboard

1. Highlight the MISQL tag. (Select only the opening MISQL tag itself, not the **IMG** dynamic tag or the closing MISQL tag.)
2. Press CTRL-C to copy the tag to the clipboard.

Now, create the new user dynamic tag, paste the contents of the clipboard, and enter the remainder of the source code.

To complete the dynamic tag

1. Choose **File→New**.
2. Press CTRL-V to paste the contents of the clipboard.
3. Type the remaining source code:

UWC

```
<IMG
  SRC=$WEB_HOME?MIvalObj=@NAME@&MItypeObj=$1
  ISMAP
  ALT="$(IF,$(EQ,@ALT@,),$4,@ALT@)">
<?/misql>
```

♦

WB

```
<IMG
  SRC=$WEB_HOME?LO=$1&MItypeObj=$2
  ISMAP
  ALT="$(IF,$(EQ,@ALT@,),$5,@ALT@)">
<?/misql>
```

♦

4. Choose **File→Save as DB.**
5. Select **Gary** from the **Project** drop-down list.
6. Type `imgmap.tag` as the AppPage name.
7. Click **OK**.
8. Type the description (`image map dynamic tag for GaryÕs Web application`) and parameters (`alt`).
9. Click **OK**.

You have created the three base dynamic tags that you need for the application.

Developing the Application

After you have created the project and dynamic tags, and inserted the binaries, you are ready to build the application. The remainder of this chapter is a narrative with a header indicating the introduction of a new Data Director for Web feature. Unless otherwise speciÞed, the remaining application development is done in AppPage Editor.

The Þrst thing you do is build the home page. The home AppPage contains a header, footer, corporate logo, and a site map to the other portions of the application.

Inserting a Dynamic Tag

There are two ways to insert a dynamic tag:

- Choose **Insert→Dynamic Tag**.
- Click the **Dynamic Tag** button on the toolbar.

To insert the header dynamic tag using the menu option

1. Choose **Insert→Dynamic Tag**.
2. Select the project **Gary** from the **Project** drop-down list.
3. Select **header** from the list box.
4. Click **OK**.

Edit the tag to include the title parameter:

```
<?header title="Gary's Gourmet Health Food">
```

To insert the footer dynamic tag using the toolbar button

1. Ensure that the cursor insertion point is after the **header** dynamic tag.
2. Click the **Dynamic Tag** button on the toolbar.
3. Select **footer**.
4. Click **OK**.

Next, you embed the corporate logo into the AppPage.

Inserting an Image

After you have inserted the dynamic tags, you must embed the corporate logo, **carrots.gif**, that you inserted into the project previously.

To embed the logo

1. Ensure that the cursor insertion point is after the **header** dynamic tag.
2. Choose **Insert→Embed Image**.
3. Select project **Gary** from the **Project** drop-down list.
4. Select **carrots**.
5. Click **OK**.
6. Ensure that the **Embed Resource** and **Dynamic Tag** options are selected.
7. Click **Insert Dynamic Tag**.

A **wbImg** dynamic tag has been inserted into the AppPage for the **carrots** resource.

Next, you create the site map to the other areas of the Web site.

Using the Query Wizard

Information about the AppPages that make up different areas of the Web site is stored in a base table called **site_map**. You use the Query wizard to build an MISQL tag to retrieve and display the site map data on the home page.

To build the query for the site map

1. Click the **Query wizard** button on the toolbar.
2. Click **Next**.
3. Select **Insert query text into current page (no tag)** as your destination. When you choose not to create a dynamic tag, no dynamic tag name is required.
4. Click **Next**.

5. Retain **HTML table** as output type but unselect **Use Headings**.
6. Click **Next**.
7. Select **site_map** as the table and click > to add it to your selected table list.
8. Click **Next**.
9. Select all columns by clicking the **site_map** table icon, and clicking >.
10. Select the **title** column by expanding **site_map**, selecting **title**, and clicking >.
11. Click **Next**.
12. Toggle **Display** to FALSE for **page_id**, **relative_weight,** and the second instance of **title** by clicking the **Display** cell for each.
13. Click **OrderBy** for **relative_weight** to use it as the sort order criterion.
14. Click **Next**.
15. There are no WHERE or HAVING clauses, so click **Next**.
16. You do not create a link, so click **Finish**.

The Query wizard has created the HTML to retrieve the site map information. Because you are passing **title** in a URL, you must edit the SQL to have the second instance of **title** retrieved using the URLENCODE variable processing function. Edit the SQL as follows:

```
<TABLE BORDER=0 CELL_PADDING=0>
<?MISQL SQL="SELECT
    site_map.title,
    site_map.page_id,
    site_map.relative_weight,
    site_map.title
FROM site_map
ORDER BY site_map.relative_weight Asc;">
<TR>
    <TD> <A
HREF="$WEB_HOME?MIval=$2&title=$(URLENCODE,$4)">$1</A> </TD>
</TR><?/MISQL>
</TABLE>
```

To save the AppPage

1. Choose **File→Save as DB**.

2. Type `/gary/home/index.html` as the AppPage name.
3. Click **OK**.

You have completed the Þrst AppPage in GaryÕs Web application. You can preview the AppPage by choosing **Tools→Preview in External Browser**.

After you have a base page, edit the **footer** dynamic tag to include a link back to the home page, as described next.

Linking an AppPage

Open the **footer** dynamic tag in the AppPage Editor, as you did in ÒCreating and Editing User Dynamic TagsÓ on page 5-6. Edit the dynamic tag to include a link back to the home page.

To edit the dynamic tag

1. Ensure that the insertion point is before the </BODY> tag.
2. Choose **Insert→Link to→AppPage**.
3. Click **/gary/home/index.html**.
4. Click **OK**.
5. Ensure that the **Link to Resource** and **Dynamic Tag** options are selected.
6. Edit the dynamic tag to include the parameter:

```
TEXT="Return to GaryÕs Gourmet Home Page"
```

7. Click **Insert tag**.
8. Choose **File→Save** to save the dynamic tag.

The **footer** dynamic tag includes a link back to the home page.

Next, create the AppPage associated with store location. You use variable processing functions to implement a simple mechanism for determining the clickable areas in the graphic.

First, create the new AppPage, and insert the **header** and **footer** dynamic tags.

Gary has four stores nationwide. This conveniently divides the image map into quadrants. The center of the image is at coordinate `(309,210)`. Use this coordinate to build a series of MIBLOCK conditions with variable processing functions.

Inserting AppPage Tags

DeÞne the image center and default region using MIVAR tags.

To create the x_center MIVAR tag

1. Click the **Insert MIVAR** button on the toolbar.
2. Highlight **NAME** under Attributes.
3. Type `x_center` in the **Value** Þeld.
4. Click **OK**.

To create the y_center MIVAR tag

1. Choose **Insert→MIVAR tag**.
2. Highlight **NAME** under Attributes.
3. Type `y_center` in the **Value** Þeld.
4. Click **OK**.

Enter the **region_chosen** MIVAR tag using the method of your choice. Edit the three tags as follows:

```
<?MIVAR NAME="x_center">309<?/MIVAR>
<?MIVAR NAME="y_center">210<?/MIVAR>
<?MIVAR NAME="region_chosen">1<?/MIVAR>
```

Having set the variables, you can begin to build MIBLOCK conditions. Build the Þrst MIBLOCK tag using the **Insert MIBLOCK** button on the toolbar and entering the variable processing function manually.

To build the MIBLOCK tag

1. Click the **Insert MIBLOCK** button on the toolbar.
2. Highlight **COND** under Attributes.
3. Type `$(XST,$x_value)` in the **Value** Þeld.

4. Click **OK**.

Build the MIBLOCK condition using variable processing functions, as follows.

Inserting Variable Processing Functions

Insert an MIBLOCK tag as described in the preceding section. Edit the MIBLOCK tag to add the COND parameter and insert the variable processing functions.

To edit the MIBLOCK tag

1. Click the **Insert VPF** button on the toolbar.
2. Select **AND**.
3. Click **Insert**.
4. Highlight *val1* in `$(AND,val1,val2)`.
5. Choose **Insert→Variable Processing Function**.
6. Select **<**.
7. Click **Insert**.
8. Highlight *val2* in `$(AND,val1,val2)`.
9. Press CTRL-I.
10. Select **>=**.
11. Click **Insert**.

Edit the MIBLOCK tag to the following:

```
<?MIBLOCK COND="$(AND,$(<,$x_value,$x_center),$(>=,$y_value,$y_center))">
```

Using AppPage Editor features previously described in ÒUsing the Query Wizard,Ó ÒInserting AppPage Tags,Ó and ÒInserting a Dynamic Tag,Ó repeat the process of inserting MIBLOCK tags, MIVAR tags, and variable processing functions to create the following AppPage segment:

```
<?MIBLOCK COND="$(XST,$x_value)">
<?MIBLOCK COND="$(AND,$(<,$x_value,$x_center),$(>=,$y_value,$y_center))">
<?MIVAR NAME=region_chosen>2<?/MIVAR>
<?/MIBLOCK>
<?MIBLOCK COND="$(AND,$(>=,$x_value,$x_center),$(<,$y_value,$y_center))">
<?MIVAR>$(SETVAR,region_chosen,4)<?/MIVAR>
<?/MIBLOCK>
<?MIBLOCK COND="$(AND,$(>=,$x_value,$x_center),$(>=,$y_value,$y_center))">
<?MIVAR NAME=region_chosen>3<?/MIVAR>
<?/MIBLOCK>
```

Create the remainder of the AppPage following the text you have already created, and save it in the database as **/gary/locations/locate.html**:

```
The closest Gary's Gourmet is located at:<P><H4>
<?MISQL SQL="select address from locations where
region=$region_chosen;">$1<?/MISQL></H4>
<P>
<HR>
<?MIVAR>$(UNSETVAR,$x_value)<?/MIVAR>
<?MIVAR>$(UNSETVAR,$y_value)<?/MIVAR>
<?/MIBLOCK>
<P>
<H3>Click on the state to find the location of the nearest store!</H3>
<P>
<CENTER>
<A
HREF="<?MIVAR>$WEB_HOME<?/MIVAR>?MImap=on&MIval=/gary/locations/locate.html&titl
e=The+Location+Nearest+You">
<?imgmap name="usa" path="/gary/images" ext="gif" ALT="Map">
</A>
</CENTER>
```

WB

To pass image map coordinates within a Web DataBlade module application in the ANCHOR tag, modify the tag to include a `/` instead of a `?` in the URL immediately preceding the MImap variable.♦

Inserting Option Lists

The next group of AppPages update grocery inventory. You insert radio button lists to update the appropriate items.

To create a radio button list

1. Click the **Option List wizard** button on toolbar.
2. Click **Next**.
3. Type `name_input` as the Name of Form Element.
4. Select the **Radio Button List** Type of Tag.
5. Click **Next**.
6. Type `select name from foods;` in **SQL for List**.
7. Leave the other Þelds blank.
8. Click **Finish**.

Use previously described AppPage Editor features to enter the rest of the following HTML:

```
<?header title="Update Available Groceries">
<H2>Update an Available Grocery Item</H2>
<?MIVAR COND=$(XST,$name_input)>$(UNSETVAR,$name_input)<?/MIVAR>
<FORM METHOD=POST ACTION=<?mivar>$WEB_HOME<?/mivar>>
<INPUT TYPE=hidden NAME=MIval VALUE="/gary/groceries/update_foods.html">
<?RADIOLIST SQL="select name from foods;" NAME="name_input">
<INPUT TYPE="submit">
<INPUT TYPE="reset">
</FORM>
<CENTER><H2>- OR -</H2></CENTER>
<H2>Delete an Available Grocery Item</H2>
<FORM METHOD=POST ACTION=<?mivar>$WEB_HOME<?/mivar>>
<INPUT TYPE=hidden NAME=MIval VALUE="/gary/groceries/delete_foods.html">
<?RADIOLIST SQL="select name from foods;" NAME="name_input">
<INPUT TYPE="submit">
<INPUT TYPE="reset">
</FORM>

<A HREF="<?MIVAR>$WEB_HOME<?/MIVAR>?MIval=/gary/groceries/addfoods.html">
Add a new food
</A>
<?footer>
```

Save the AppPage as **/gary/groceries/foods.html**.

Using the Update Form Wizard

Create a new page to update grocery inventory. First, insert header and footer dynamic tags and save the page as **/gary/groceries/update_foods.html**. Use the Update wizard to create two dynamic tags, **grocery_edit** and **grocery_update**.

To create the grocery dynamic tags

1. Click the **Update wizard** button on toolbar.
2. Click **Next**.
3. Type `grocery` as Name of new dynamic tag.
4. Type `/gary/groceries/update_foods.html` as AppPage to POST.
5. Select the **Insert both tags into current page** option.
6. Click **Next**.
7. Select **foods** as the target table.
8. Toggle **Use as Key** to `TRUE` for **name**.
9. Click **Next**.
10. Accept the default success message by clicking **Next**.
11. Accept the default tag setting by clicking **Finish**.

You have created the dynamic tags and saved them in the database. Save the page with the new dynamic tags inserted by clicking the **Save** button on the toolbar.

Using the Delete Wizard

Create a new page to delete groceries from the inventory. First, insert header and footer dynamic tags. Use the Delete wizard to create the MISQL tag to perform the delete.

To use the Delete wizard

1. Click the **Delete wizard** button on the toolbar.
2. Click **Next**.
3. Select the **Insert text into current page** option.
4. Click **Next**.

5. Select **foods** as the target table.
6. Toggle **Use as Key** to `TRUE` for **name**.
7. Click **Next**.
8. Accept the default success message by clicking **Finish**.

You have created the MISQL tag and inserted it into the AppPage. Save the page as **/gary/groceries/delete_foods.html**.

Using the Input Form Wizard

Create a new page to add groceries to the inventory list. First, insert header and footer dynamic tags and save the page as **/gary/groceries/addfoods.html**. Use the Insert wizard to create two dynamic tags, **grocery_insert** and **grocery_add**.

To use the Input Form Wizard

1. Click the **Input Form wizard** button on the toolbar.
2. Click **Next**.
3. Type `grocery` as the name of the new dynamic tag.
4. Type `/gary/groceries/addfoods.html` as the AppPage to POST.
5. Select the **Insert both tags into current page** option.
6. Click **Next**.
7. Select **foods** as the target table.
8. Accept all Þelds as mandatory input Þelds by clicking **Next**.
9. Accept the default success message by clicking **Next**.
10. Accept the default tag setting by clicking **Finish**.

You have created the dynamic tags and saved them in the database. Save the page with the new tags inserted by clicking the **Save** button on the toolbar.

Now That You Have Finished

After you have completed building the sample application, you can access it by invoking the URL:

UWC

```
http://webserver/ifmx-
bin/wcdrvr.exe/wbdaemon.exe?MIval=/gary/home/index.html ♦
```

WB

```
http://webserver/cgi-
bin/webdriver?MIval=/gary/home/index.html ♦
```

Chapter

Deploying a Web Application Using Data Director for Web

6

In This Chapter

Deployment is the process by which you copy a Web application from one Data Director for Web installation to another. You might, for instance, deploy a newly developed Web application from your dynamic development environment to a more controlled test environment and, Þnally, to your stable production environment.

This chapter explains how to use Site Manager to deploy a Web application from the Deployment dialog box or from the Export/Import dialog boxes.

You must use the Export/Import option when copying a Web application between installations on computers that are not networked. Otherwise, you can use either option.

Deployment Dialog Box

There are three ways to access the Deployment dialog box:

- From the **Tools→Deployment** menu option
- From the **Deployment** button on the toolbar
- By dragging the project from the source database to the target database

Using the drop-down menus in the dialog box, supply for the deployment source and target:

- the name of the database server, as registered in **setnet32**.
- the name of the database, with the Data Director for Web schema installed.
- the name of the project.

Your source and destination databases must be distinct; you cannot deploy an application within the source database. In addition, you must create the target project prior to deploying the application.

Data Director for Web tracks the versions of a resource when it is deployed, and the date and time that it was deployed. You inspect the status of deployed databases and their component resources using the **Edit→Properties** menu option.

There are two deployment methods: *full deployment* and *incremental deployment*.

Under full deployment, all resources from the source project are copied to the target database, even if an existing target resource is a newer version than the source resource. Use this option to force deployment of a complete Web application when you are certain that no resources will conßict with other applications in the same database. Full deployment is the default option.

Important: If a resource already exists in the target database, it is automatically overwritten with the deployed resource.

Under incremental deployment, a resource is copied from the source project to the target project only if it is a newer version than the target resource. Incremental deployment is generally faster than full deployment because fewer Þles are copied. Use this option when updating a Web application, or if you are concerned that resources might conßict.

Export/Import Dialog Boxes

There are two ways to access the Export/Import dialog boxes:

- From the **Tools→Export** and **Tools→Import** menu options
- From the **Export** and **Import** buttons on the toolbar

Using the latter method, export and import are distinct steps.

Export

Using the Export dialog box, supply:

- the root directory to which you want to copy the project resources.
- the name of the source database server, as registered in **setnet32**.
- the name of the source database in which the project is stored.
- the name of the source project in which the application is stored.

The dynamic tags are copied into the root directory; all other resources are copied into a directory structure under the root directory corresponding to the path stored in the Data Director for Web schema. For example, if the image resource **/path/name/image.gif** is exported to the root directory **/rootdir**, the Þle is copied to **/rootdir/path/name/image.gif** on the Þle system.

After exporting the project resources to the Þle system, you import them into the target database.

Import

Using the Import dialog box, supply:

- the root directory from which you want to copy the project resources.
- the name of the target database server, as deÞned in **setnet32**.
- the name of the target database in which the project is to be stored.
- the name of the target project in which the application is to be stored.

The dynamic tags are copied from the root directory; all other resources are copied into the target database and given a resource path relative to the directory structure under the root directory. For example, if the image **/rootdir/path/name/image.gif** is imported into the Data Director for Web schema, the resource name is **image.gif** and the resource path is **/path/name**.

On import, you have the option of converting Þle-system-based links to Webdriver-compatible links valid in the database. To convert the links, click the **Convert Links on Import** box in the Import dialog box. For more information on converting Þle system links, see ÒConverting File System LinksÓ on page 4-12.

Appendix

A

Data Director for Web Schema DeÞnitions

This appendix contains the schema deÞnitions for the tables in the Data Director for Web schema for both Universal Web Connect and the Web DataBlade module.

wbPages

The **wbPages** table stores AppPages and any other text format resources, such as plain text Þles or HTML pages. Resources other than AppPages have extensions that map to types not expanded by **WebExplode**.

Column	UWC Type	WB Type	Description
ID	VARCHAR(30)	VARCHAR(30)	The name of the AppPage
path	VARCHAR(178)	VARCHAR(178)	The Þle system path to the AppPage
extension	VARCHAR(12)	VARCHAR(12)	The Þle system extension of the AppPage
description	VARCHAR(254)	VARCHAR(254)	The description of the AppPage
author	VARCHAR(30)	VARCHAR(30)	The author of the AppPage
keywords	VARCHAR(254)	VARCHAR(254)	The keywords associated with the AppPage
current_version	INTEGER	INTEGER	The version number of the AppPage
last_changed	DATETIME	DATETIME	When the AppPage was last modiÞed
last_changed_by	VARCHAR(30)	VARCHAR(30)	The user who last changed the AppPage
read_level	INTEGER	INTEGER	The run-time security level required to read the AppPage

(1 of 2)

Column	UWC Type	WB Type	Description
last_locked	DATETIME	DATETIME	When the AppPage was last locked
last_locked_by	VARCHAR(30)	VARCHAR(30)	The user that locked the AppPage
object	TEXT IN %s	HTML	The content of the AppPage

(2 of 2)

wbBinaries

The **wbBinaries** table stores all nontext resources.

Column	UWC Type	WB Type	Description
ID	VARCHAR(30)	VARCHAR(30)	The name of the resource
path	VARCHAR(178)	VARCHAR(178)	The Þle system path to the resource
extension	VARCHAR(12)	VARCHAR(12)	The Þle system extension of the resource
description	VARCHAR(254)	VARCHAR(254)	The description of the resource
height	INTEGER	INTEGER	The height of the resource, used to build an IMG or EMBED tag
width	INTEGER	INTEGER	The width of the resource, used to build an IMG or EMBED tag
current_version	INTEGER	INTEGER	The version number of the resource
last_changed	DATETIME	DATETIME	When the resource was last modiÞed
last_changed_by	VARCHAR(30)	VARCHAR(30)	The user who last changed the resource
read_level	INTEGER	INTEGER	The run-time security level required to read the resource

(1 of 2)

Column	UWC Type	WB Type	Description
last_locked	DATETIME	DATETIME	When the resource was last locked
last_locked_by	VARCHAR(30)	VARCHAR(30)	The user who locked the resource
object	BYTE IN %s	BLOB	The content of the resource

(2 of 2)

wbTags

The **wbTags** table stores Web Integration Option dynamic tags.

Column	UWC Type	WB Type	Description
ID	VARCHAR(30)	VARCHAR(30)	The name of the tag
description	VARCHAR(254)	VARCHAR(254)	The description of the tag
parameters	VARCHAR(254)	VARCHAR(254)	The parameters passed to the tag
class	VARCHAR(64)	VARCHAR(64)	The class of the tag
current_version	INTEGER	INTEGER	The version number of the tag
last_changed	DATETIME	DATETIME	When the tag was last modiÞed
last_changed_by	VARCHAR(30)	VARCHAR(30)	The user who last changed the tag
last_locked	DATETIME	DATETIME	When the tag was last locked
last_locked_by	VARCHAR(30)	VARCHAR(30)	The user that locked the tag
object	TEXT IN %s	HTML	The content of the tag
wizData	BYTE IN %s	BLOB	The properties of the tag(wizard-generated tags only)

wbExtensions

The **wbExtensions** table stores type-mapping information.

Column	UWC Type	WB Type	Description
extension	VARCHAR(12)	VARCHAR(12)	The Þle extension
name	VARCHAR(30)	VARCHAR(30)	The name of the extension
source_table	VARCHAR(18)	VARCHAR(18)	The name of the table in which the resource is stored
super_type	VARCHAR(18)	VARCHAR(18)	The MIME supertype of the extension
sub_type	VARCHAR(18)	VARCHAR(18)	The MIME subtype of the extension
ID_column	VARCHAR(18)	VARCHAR(18)	The name of the column containing the resource identiÞer
content_column	VARCHAR(18)	VARCHAR(18)	The name of the column containing the resource content
path_column	VARCHAR(18)	VARCHAR(18)	The name of the column containing the resource path information
retrieval_method	INTEGER	INTEGER	The retrieval method used by Webdriver when retrieving the type

wbProjects

The **wbProjects** table stores the project information for all projects in the database.

Column	UWC Type	WB Type	Description
name	VARCHAR(30)	VARCHAR(30)	The name of the project
description	VARCHAR(254)	VARCHAR(254)	The description of the project
owner	VARCHAR(30)	VARCHAR(30)	The owner of the project
last_locked	DATETIME	DATETIME	When the project was last locked
last_locked_by	VARCHAR(30)	VARCHAR(30)	The user that locked the project
last_deployed	DATETIME	DATETIME	When the project was last deployed
last_deployed_by	VARCHAR(30)	VARCHAR(30)	The user who last deployed the project
deployed_db	VARCHAR(18)	VARCHAR(18)	The database from which the project was last deployed
deployed_server	VARCHAR(18)	VARCHAR(18)	The database server from which the project was last deployed
deployed_project	VARCHAR(30)	VARCHAR(30)	The name of the project from which this project was last deployed
preview	VARCHAR(30)	VARCHAR(30)	The preview conÞguration name for the project

wbResProjects

A project can have many resources associated with it, and a resource can belong to many projects. The **wbResProjects** table manages the many-to-many relationship between projects and resources.

Column	UWC Type	WB Type	Description
project_name	VARCHAR(30)	VARCHAR(30)	The name of the project
ID	VARCHAR(30)	VARCHAR(30)	The name of the resource
path	VARCHAR(178)	VARCHAR(178)	The path of the resource
extension	VARCHAR(12)	VARCHAR(12)	The extension of the resource

wbPreviews

The **wbPreviews** table contains the preview conÞgurations for each project.

Column	UWC Type	WB Type	Description
ID	VARCHAR(30)	VARCHAR(30)	The name of the preview conÞguration
object	TEXT IN %s	HTML	The preview conÞguration settings

Appendix

B

Data Director for Web Hot Keys and Shortcuts

The following hot keys and shortcuts are available in AppPage Editor only, unless otherwise speciÞed.

Category	Key Sequence	Action
Edit	CTRL-C	Copy to clipboard (also available in Site Manager)
	CTRL-F	Find
	CTRL-G	Go to
	CTRL-I	Insert variable processing function
	CTRL-R	Replace
	CTRL-V	Paste from clipboard (also available in Site Manager)
	CTRL-X	Cut to clipboard
	F3	Find next
File	CTRL-N	New
	CTRL-O	Open
	CTRL-P	Print
	CTRL-S	Save
Help	F1	Invoke help

(1 of 2)

Category	Key Sequence	Action
Navigate	F5	Refresh / Reload
	ESC	Stop
	CTRL-B	Find closing </?MIBLOCK> tag for selected <?MIBLOCK> tag
View	CTRL-P	Toggle Edit / Preview modes

(2 of 2)

In addition, in AppPage Editor, the **File** menu displays the four most recently accessed database resources and the four most recently accessed Þles.

Appendix C

Data Director for Web Example Application

This appendix contains the complete code from Chapter 5, ÒBuilding an Application with Data Director for Web.Ó You can also Þnd these examples in the following directories:

UWC **$INFORMIXDIR\ddw\Examples\Non-UDO\gary** ♦

WB **$INFORMIXDIR\ddw\Examples\UDO\gary** ♦

The listings in this appendix contain the source code for the Universal Web Connect examples. In addition to the differences between the Universal Web Connect and Web DataBlade module examples highlighted in Chapter 5, the Web DataBlade module examples retrieve large objects by handle instead of by name. Retrieving large objects by handle allows you to take advantage of Web DataBlade module large object caching. See the *Informix Web DataBlade Module UserÕs Guide* for information on retrieving large objects by handle and caching large objects.

To install the example application into your database, you import the resources into your project and create and populate the data tables associated with the application.

To install the example application

1. Create a project as described in ÒCreating a ProjectÓ on page 5-4.
2. Open Windows Explorer or File Manager.
3. Open either the **Non-UDO** or **UDO** directory under the **Examples** directory. Click the **gary** directory and drag it into the project you created in Site Manager.
4. Site Manager prompts ÒDo you want to convert links on import?Ó Click **OK**.

5. Site Manager reports that three SQL Þles were not imported because their extensions are unknown. This is expected behavior.
6. Execute the scripts in the **gary\sql** directory in **SQL Editor** or **dbaccess** by connecting to the database and opening and executing each Þle ending with the **.sql** extension.

You can also add the examples to your database by importing them using the **Import** utility in the **Tools** menu of Site Manager. However, if you use this method, the paths in the links are not converted to the relative paths of the resources that are imported. After importing, you must modify the path for each resource.

You can also copy and paste individual code segments to add one or more resources.

Schema

Following is the SQL to create the schema for the example application.

sitemap.sql

```
create table site_map(title varchar(50),
             page_id varchar(75),
             relative_weight integer);
insert into site_map values ("Locate a Store",
                "/gary/locations/locate.html",
                100);
insert into site_map values ("Update Available Grocery Items",
                             "/gary/groceries/foods.html",
                             200);
```

locations.sql

```
create table locations (region integer, address varchar(50));
insert into locations values (1, 'Peets Plaza, Seattle, Washington');
insert into locations values (2, 'Avenue of the Stars, Hollywood, California');
insert into locations values (3, 'Middle of the Everglades, Florida');
insert into locations values (4, 'Bicentennial Way, Boston, Massachusetts');
```

foods.sql

```
create table foods(name varchar(20), qty_avail integer, vegetarian boolean);
insert into foods values ('broccoli',1000,'t');
insert into foods values ('mushrooms',50000,'t');
insert into foods values ('carrots',10,'t');
insert into foods values ('caribou',1,'f');
insert into foods values ('cheddar cheese',100,'t');
insert into foods values ('garlic',1000,'t');
insert into foods values ('potatoes',0,'t');
insert into foods values ('noodles',100000,'t');
insert into foods values ('sour cream',1000,'t');
insert into foods values ('chocolate',25000,'t');
insert into foods values ('vanilla ice cream',1000,'t');
insert into foods values ('strawberries',50,'t');
```

AppPages

The AppPages are arranged approximately in the order in which they are invoked.

/wb/gary/home/index.html

```
<!-- begin index1 -->
<?header title="Gary's Gourmet Health Food">
<!-- end index1 -->
<CENTER>
<?wbImg name="carrots" path="/gary/images" ext="gif"
alt="corporate logo" >
<H1>Welcome to Gary's Gourmet Health Food To-Go!</H1>
<!-- begin index2 -->
<TABLE BORDER=0 CELL_PADDING=0>
<?MISQL SQL="SELECT
    site_map.title,
    site_map.page_id,
    site_map.relative_weight,
    site_map.title
FROM site_map
ORDER BY site_map.relative_weight Asc;">
<TR>
    <TD> <A
HREF="$WEB_HOME?MIval=$2&title=$(URLENCODE,$4)">$1</A> </TD>
</TR><?/MISQL>
</TABLE>
<!-- end index2 -->
</CENTER>
<?footer>
```

/wb/gary/locations/locate.html

```
<?header title="<?MIVAR>$title<?/MIVAR>">
<!-- begin locate1 -->
<?MIVAR NAME="x_center">309<?/MIVAR>
<?MIVAR NAME="y_center">210<?/MIVAR>
<?MIVAR NAME="region_chosen">1<?/MIVAR>
<!-- end locate1 -->
<P>
<!-- begin locate2 -->
<?MIBLOCK COND="$(XST,$x_value)">
<?MIBLOCK
COND="$(AND,$(<,$x_value,$x_center),$(>=,$y_value,$y_center)
)">
<?MIVAR NAME=region_chosen>2<?/MIVAR>
<?/MIBLOCK>
<?MIBLOCK
COND="$(AND,$(>=,$x_value,$x_center),$(<,$y_value,$y_center)
)">
<?MIVAR>$(SETVAR,region_chosen,4)<?/MIVAR>
<?/MIBLOCK>
<?MIBLOCK
COND="$(AND,$(>=,$x_value,$x_center),$(>=,$y_value,$y_center
))">
<?MIVAR NAME=region_chosen>3<?/MIVAR>
<?/MIBLOCK>
<!-- end locate2 -->
<!-- begin locate3 -->
The closest Gary’s Gourmet is located at:<P><H4>
<?MISQL SQL="select address from locations where
region=$region_chosen;">$1<?/MISQL></H4>
<P>
<HR>
<?MIVAR>$(UNSETVAR,$x_value)<?/MIVAR>
<?MIVAR>$(UNSETVAR,$y_value)<?/MIVAR>
<?/MIBLOCK>
<P>
<H3>Click on the state to find the location of the nearest
store!</H3>
<P>
<CENTER>
<A
HREF="<?MIVAR>$WEB_HOME<?/MIVAR>?MImap=on&MIval=/gary/locati
ons/locate.html&title=The+Location+Nearest+You">
<?imgmap name="usa" path="/gary/images" ext="gif" ALT="Map">
</A>
</CENTER>
<!-- end locate3 -->
<?footer>
```

/wb/gary/groceries/foods.html

```
<?header title="Update Available Groceries">
<H2>Update an Available Grocery Item</H2>
<?MIVAR
COND=$(XST,$name_input)>$(UNSETVAR,$name_input)<?/MIVAR>
<FORM METHOD=POST ACTION=<?mivar>$WEB_HOME<?/mivar>>
<INPUT TYPE=hidden NAME=MIval
VALUE="/gary/groceries/update_foods.html">
<?RADIOLIST SQL="select name from foods;" NAME="name_input">
<INPUT TYPE="submit">
<INPUT TYPE="reset">
</FORM>
<CENTER><H2>- OR -</H2></CENTER>
<H2>Delete an Available Grocery Item</H2>
<FORM METHOD=POST ACTION=<?mivar>$WEB_HOME<?/mivar>>
<INPUT TYPE=hidden NAME=MIval
VALUE="/gary/groceries/delete_foods.html">
<?RADIOLIST SQL="select name from foods;" NAME="name_input">
<INPUT TYPE="submit">
<INPUT TYPE="reset">
</FORM>

<A
HREF="<?MIVAR>$WEB_HOME<?/MIVAR>?MIval=/gary/groceries/addfo
ods.html">
Add a new food
</A>
<?footer>
```

/wb/gary/groceries/update_foods.html

```
<?header title="Update grocery availability">
<?grocery_update>
<?grocery_edit>
<?footer>
```

/wb/gary/groceries/delete_foods.html

```
<?header title="Delete a food">
<?misql sql="delete from foods where name = '$name_input';">
Success! $MI_ROWCOUNT row(s) deleted.<?/misql>
<?footer>
```

/wb/gary/groceries/addfoods.html

```
<?header title="Add foods">
<?grocery_add>
<?grocery_insert>
<?footer>
```

Tags

Tags are presented in alphabetic order, except where wizard-generated tags are displayed as pairs.

footer

```
<!-- begin footer1 -->
<P>
<HR>
<P>
<CENTER>Many thanks to our sponsors:<BR>

Harry's House of Hummus |
<A HREF="http://www.informix.com">Informix Corp.</A>
<P>
</CENTER>
<!-- end footer1 -->
<!-- begin footer2 -->
<?wbLink name="index" path="/gary/home" ext="html"
        TEXT="Return to Gary's Gourmet Home Page">
<!-- end footer2 -->
<!-- begin footer3 -->
</BODY>
</HTML>
<!-- end footer3 -->
```

grocery_add

```
<!-- (IWB APE) Database insert form for the foods table.
Generated by INFORMIX IWB APE, Tuesday 11:59:37, August 19,
1997, by strother-->
<!-- (IWB APE) Setup state variable -->
<?mivar name="apb_state_"
default="GET">$apb_state_<?/mivar>
<?mivar name="apb_error_"><?/mivar>
<!-- (IWB APE) POST state: process the input to the table -->
<?miblock cond=$(EQ,$apb_state_,POST)>

    <!-- (IWB APE) validate mandatory fields -->

    <?miblock cond=$(EQ,$apb_state_,POST)>
    <!-- (IWB APE) assign default values -->
        <?mivar name="apb_name"
default="">$apb_name<?/mivar>
        <?mivar name="apb_qty_avail"
default="0">$(IF,$(EQ,$apb_qty_avail,),"0",$apb_qty_avail)<?
/mivar>
        <?mivar name="apb_vegetarian"
default="">$apb_vegetarian<?/mivar>

        <!-- (IWB APE) validate that integers are integers -->
        <?mivar name="apb_state_"
cond=$(NOT,$(ISNUM,$apb_qty_avail))>ERROR<?/mivar>
        <?mivar name="apb_error_"
cond=$(NOT,$(ISNUM,$apb_qty_avail))>Field "qty_avail" was an
invalid numeric.<?/mivar>

        <!-- (IWB APE) replace ' with double '' so sql will
not fail -->
        <?mivar
name="apb_name">$(REPLACE,$apb_name,',''')<?/mivar>
        <?mivar
name="apb_qty_avail">$(REPLACE,$apb_qty_avail,',''')<?/mivar>
        <?mivar
name="apb_vegetarian">$(REPLACE,$apb_vegetarian,',''')<?/miva
r>

        <!-- (IWB APE) Build SQL for insert statement -->
        <?mivar name="apb_sql")>
        INSERT INTO foods (
        name, qty_avail, vegetarian) VALUES (
        '$apb_name', $apb_qty_avail,
'$apb_vegetarian');<?/mivar>
        <!-- (IWB APE) Process Insert SQL if no errors -->
        <?misql cond=$(EQ,$apb_state_,POST)
sql="$apb_sql">Success! $MI_ROWCOUNT rows inserted. <?/misql>
```

```
    <?/miblock>
    <!-- (IWB APE) Display error message if we got an error -->
    <?mivar
cond=$(EQ,$apb_state_,ERROR)>$apb_error_<?/mivar>

    <!-- (IWB APE) Reset State so form is redisplayed for
errors -->
    <?mivar name="apb_state_"
cond=$(EQ,$apb_state_,ERROR)>GET<?/mivar>
    <?/miblock>
```

grocery_insert

```
<!-- (IWB APE) Database insert form for the foods table.
Generated by INFORMIX IWB APE, Tuesday 11:59:37, August 19,
1997, by strother-->
<!-- (IWB APE) Setup state variable -->
<?mivar name=apb_state_ default="GET">$apb_state_<?/mivar>
<!-- (IWB APE) GET State: display the HTML form for the table
-->
<?miblock cond=$(EQ,$apb_state_,GET)>
    <!-- (IWB APE) input form -->
    <FORM METHOD=POST ACTION=<?mivar>$WEB_HOME<?/mivar>
    <!-- (IWB APE) hidden variables -->
    <INPUT TYPE=hidden NAME=MIval
VALUE="/gary/groceries/addfoods.html">
    <INPUT TYPE=hidden NAME=apb_state_ VALUE="POST">
        <TABLE BORDER=1>
    <?mivar name="apb_name" default="">$apb_name<?/mivar>
    <?mivar name="apb_qty_avail"
default="">$apb_qty_avail<?/mivar>
    <?mivar name="apb_vegetarian"
default="">$apb_vegetarian<?/mivar>

    <?mivar>
    <tr>
    <td><b> name <b>:</td>
    <td><input type=text name="apb_name" value="$apb_name"
size="20"> (varchar) <BR></td>
    </tr>
    <tr>
    <td><b> qty_avail <b>:</td>
    <td><input type=text name="apb_qty_avail"
value="$apb_qty_avail" size="12"> (int) <BR></td>
    </tr>
    <tr>
    <td><b> vegetarian <b>:</td>
    <td><input type=text name="apb_vegetarian"
value="$apb_vegetarian" size="1"> (char) <BR></td>
    </tr>
    <?/mivar>
    </TABLE><P>
    <!-- (IWB APE) Submit button for form -->
    <input type="submit" value="Submit">
    </FORM>
<?/miblock>
<?MIVAR>$(UNSETVAR,$apb_state_)<?/MIVAR>
<?MIVAR>$(UNSETVAR,$apb_name)<?/MIVAR>
<?MIVAR>$(UNSETVAR,$apb_qty_avail)<?/MIVAR>
<?MIVAR>$(UNSETVAR,$apb_vegetarian)<?/MIVAR>
```

grocery_edit

```
<!-- (IWB APE) Database update form for the foods table.
 Generated by INFORMIX WB APE, Tuesday 11:50:24, August 19,
1997, strother -->
<!-- (IWB APE) Setup state variable -->
<?mivar name="apb_state_" default="GET">$apb_state_<?/mivar>
<!-- (IWB APE) GET state: display the HTML form for the table
-->
<?miblock cond=$(EQ,$apb_state_,GET)>
    <!--- (IWB APE) update form -->
    <FORM METHOD=POST ACTION=<?mivar>$WEB_HOME<?/mivar>
    <!-- (IWB APE) Hidden variables -->
    <INPUT TYPE=hidden NAME=MIval
VALUE="/gary/groceries/update_foods.html">
    <INPUT TYPE=hidden NAME=apb_state_ VALUE="POST">
    <H3> Update Form for table foods</H3>

    <!-- (IWB APE) Load Values for current row -->
    <?MIBLOCK COND=$(NXST,$apb_name)>
        <?MIVAR NAME="apb_name"><?/MIVAR>
        <?MIVAR NAME="apb_qty_avail"><?/MIVAR>
        <?MIVAR NAME="apb_vegetarian"><?/MIVAR>
        <?misql sql="SELECT name, qty_avail, vegetarian FROM
foods
        WHERE name = '$name_input';">
        $(SETVAR,$apb_name,$1)
        $(SETVAR,$apb_qty_avail,$2)
        $(SETVAR,$apb_vegetarian,$3)
      <?/misql>
    <?/MIBLOCK>
    <?mivar>
    <TABLE>
    <tr>
        <td><b> name: </b> </td>
        <td>$apb_name <I>(key value)</i>  <INPUT type=hidden
name="apb_name" value="$apb_name"></td>
    </tr>
    <tr>
        <td><b> qty_avail: </b> </td>
        <td> <input type=text name="apb_qty_avail"
value="$apb_qty_avail" size="8">  </td>
    </tr>
    <tr>
        <td><b> vegetarian: </b> </td>
        <td> <input type=text name="apb_vegetarian"
value="$apb_vegetarian" size="1">  </td>
    </tr>
    </TABLE>
```

```
    <?/mivar><P>
    <!-- (IWB APE) Submit button for form -->
    <INPUT type=submit value="Submit">
    </FORM>
<?/miblock>
<?MIVAR>$(UNSETVAR,$apb_state_)<?/MIVAR>
<?MIVAR>$(UNSETVAR,$apb_name)<?/MIVAR>
<?MIVAR>$(UNSETVAR,$apb_qty_avail)<?/MIVAR>
<?MIVAR>$(UNSETVAR,$apb_vegetarian)<?/MIVAR>
```

grocery_update

```
<!-- (IWB APE) Database update form for the foods table.
 Generated by INFORMIX WB APE, Tuesday 11:50:24, August 19,
1997, strother -->
<!-- (IWB APE) Setup state variable -->
<?mivar name="apb_state_" default="GET">$apb_state_<?/mivar>
<?mivar name="apb_error_"><?/mivar>
<!-- (IWB APE) POST state: process the update to the table -->
<?miblock cond=$(EQ,$apb_state_,POST)>

    <!-- (IWB APE)  Validate that integers are integers -->
    <?mivar name="apb_state_"
cond=$(NOT,$(ISNUM,$apb_qty_avail))>ERROR<?/mivar>
    <?mivar name="apb_error_"
cond=$(NOT,$(ISNUM,$apb_qty_avail))>Field "qty_avail" was an
invalid numeric.<?/mivar>
    <!-- (IWB APE)  replace ' with double '' so sql will not
fail -->
    <?mivar name="apb_qty_avail" default=""
>$(REPLACE,$apb_qty_avail,',''))<?/mivar>
    <?mivar name="apb_vegetarian" default=""
>$(REPLACE,$apb_vegetarian,',''))<?/mivar>
    <!-- (IWB APE)  Build SQL for update statement -->
    <?mivar name="apb_sql_"> UPDATE foods SET
    qty_avail = '$apb_qty_avail' ,
    vegetarian = '$apb_vegetarian'
    WHERE name = '$apb_name' ;<?/mivar>

    <!-- (IWB APE) Process Update SQL if no errors -->
    <?misql cond=$(EQ,$apb_state_,POST)
sql="$apb_sql_">Success!! $MI_ROWCOUNT rows updated <?/misql>

    <!-- (IWB APE)  Display Error message if we got an error
-->
    <?mivar
cond=$(EQ,$apb_state_,ERROR)>$apb_error_<?/mivar>

    <!-- (IWB APE)  Reset state so form is redisplayed on
errors -->
    <?mivar name="apb_state_"
cond=$(EQ,$apb_state_,ERROR)>GET<?/mivar>
<?/miblock>
```

header

```
<HTML>
<HEAD>
<TITLE>@title@</TITLE>
</HEAD>
<BODY>
<CENTER><EM>Low-fat is where it's at!</EM></CENTER>
<HR>
<P>
```

imgmap

```
<!-- begin imgmap1 -->
<?misql sql="SELECT b.super_type || '/' || b.sub_type,
a.width, a.height, a.description
    FROM wbBinaries a, wbExtensions b
    WHERE a.ID='@NAME@'
    AND a.path='@PATH@'
    AND a.extension='@EXT@'
    AND a.extension=b.extension;">
<!-- end imgmap1 -->
<!-- begin imgmap2 -->
<IMG
  SRC=$WEB_HOME?MIvalObj=@NAME@&MItypeObj=$1
  ISMAP
  ALT="$(IF,$(EQ,@ALT@,),$4,@ALT@)">
<?/misql>
<!-- end imgmap2 -->
```

Index

A

B

C

D

T

U

V

W