LassoSoft

Lasso Studio 8.5

For Eclipse

User Guide

Trademarks

Third Party Links

Copyright

Second Edition: August 13, 2007

Version: 1.5.2

LassoSoft, LLC
dba OmniPilot Software
P.O. Box 33
Manchester, Washington 98353
U.S.A.

Telephone: (954) 302-3526
Email: info@lassosoft.com
Web Site: http://www.lassosoft.com

Contents

1

Chapter 1
Introduction

This chapter provides an overview of Lasso Studio and the Lasso 8.5 product line, including new features, documentation, and other useful information.

- *Lasso Studio* introduces the features of Lasso Studio, and describes the available editions.
- *The Lasso 8.5 Product Line* introduces the full range of Lasso 8.5 products and how they relate to Lasso Studio.
- *Documentation* describes the documentation included with Lasso Studio.
- *Customer Support* introduces the resources available to help you set up and use Lasso Studio.
- *Usage Rights* includes important information about usage rights for Lasso Studio.

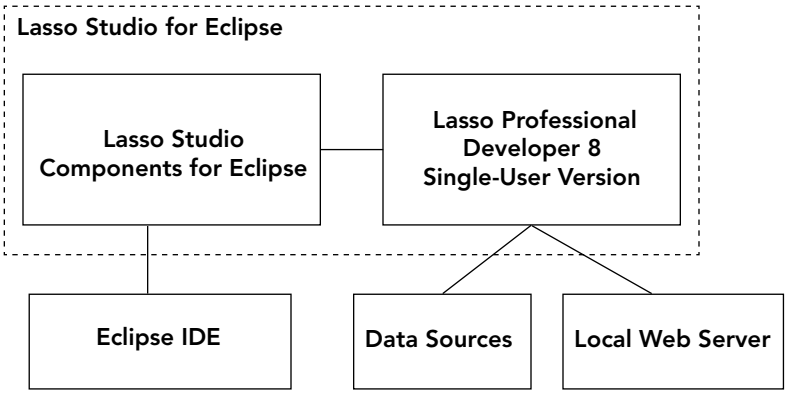
Lasso Studio

Welcome to Lasso Studio, the professional-grade authoring tool for Lasso Professional. Lasso Studio gives you the ability to create and debug powerful data-driven Web sites within the integrated development environment Eclipse. Solutions can be run directly within Eclipse and can be tested with an included single-user version of Lasso Professional Developer.

Components

Lasso Studio consists of two main components, which are the components for Eclipse and the included single-user edition of Lasso Professional Developer..

Figure 1: Lasso Studio Components



The features which Lasso Studio adds to the Eclipse IDE are described in the *Features* section below.

An included single-user license of Lasso Professional Developer allows you to test your Lasso solutions on your local computer before uploading them to a production server. This feature interfaces with the built-in Apache Web server on Mac OS X or Linux, WebSTAR V on Mac OS X, or the built-in IIS Web server on Windows, and operates separately from Eclipse..

Lasso Professional Developer included with Lasso Studio is identical to Lasso Professional Server, but goes into single-user mode when initialized with a Lasso Studio serial number. Single-user mode imposes the following restrictions over a full multi-user license of Lasso Professional Server:

- Lasso Service will only serve to the first two client IP addresses which connect via a Web browser.
- Only 500 connections (or Lasso Service actions) are allowed per minute.
- The information bar in Lasso Site Administration says Single-User instead of Multi-User.

What's New

This section describes the new features in each release of Lasso Studio for Eclipse:

Lasso Studio for Eclipse 1.5

- **Eclipse SDK 3.1** – Now supports Eclipse SDK 3.1 including performance and interface improvements. More information and downloads for the new version can be found at the Eclipse Foundation's Web site.

  http://www.eclipse.org/

- **Code Formatting** – LassoScript code can be automatically formatted to the preferred format. Options include using <?LassoScript ... ?> or square bracket [...] delimiters, using tabs or spaces for indentation, and how to handle white space and text wrap.
- **Code Completion Templates** – User customizable templates provide a guided approach to writing complex tags including placeholders for database and table names and more.
- **Variable Completion** – Variable name completion has been improved with better auto-detection of variable names.
- **Custom Type Completion** – Custom types now provide member tag and member variable completion for the [Self->...] tag.
- **HTML Results** – The Lasso Script HTML Result view now includes both a Browser View and the Source View of the results. A URL input and reload button make it easier to see what page's results are being viewed. URLs displayed within the result view can now be clicked to visit the linked page.
- **More Preferences** – New preferences allow the set of tags which are shown in the outline to be customized, a template to be specified for new Lasso Script files and for auto-completion templates, and the serial number of Lasso Studio to be easily updated.
- **Keyboard Shortcuts** – Many commands now have keyboard shortcuts including the comment toggles on the edit menu and outline view controls.
- **Encoding and Line Endings** – The Edit menu now includes commands for modifying a Lasso Script file's encoding or line endings.
- **Other Enhancements and Fixes** – Many minor improvements have been made throughout Lasso Studio. See the change notes in the Documentation folder for full details.

Features

This section describes the features of the Lasso Studio components for Eclipse. For a description of features in Lasso Professional 8.5, see the *Lasso Professional 8.5* chapter in the Lasso Setup Guide.

Lasso Studio provides the following features:

- **Syntax Checking** – LassoScripts are automatically checked for syntax errors and any problems are reported in the Problems panel.

- **Debugging** – Integrated debugger allows HTTP requests to be simulated, breakpoints to be added, and Lasso code to be stepped through line by line.
- **Formatting** – Lasso code is automatically formatted according to user preference. Existing code can be reformatted.
- **Syntax Coloring** – Lasso code is colored for ease of editing within Eclipse.
- **Code Completion** – Automatically completes Lasso tag names and includes sample syntax.
- **Code Outlining** – Provides an outline of the code within a Lasso page. The outline makes it easy to navigate through complex container tag structures.
- **Code Folding** – Container tags can be folded within the Eclipse editor allowing portions of a page to be temporarily hidden.
- **Boiler-Plating** – Provides custom tag and custom data type templates including namespaces, common options, and parameters.
- **Lasso Reference** – Provides information about each Lasso tag within the Eclipse IDE.

The Lasso 8.5 Product Line

This section provides an overview of the Lasso product line, and shows how Lasso Studio fits in. The Lasso product line consists of a set of applications and connectors that enable you to build data-driven Web sites and serve them via a Web server.

Lasso Professional Server 8

Lasso Professional Server 8 is the product for serving Lasso-based data-driven Web sites. It consists of a core Lasso Service application and several Lasso Connectors which establish links to Web servers and data sources. Lasso Professional 8.5 is available for Mac OS X and Windows. Lasso Professional 8.5 has the following components:

- **Lasso Service** – The core application or Web Data Engine. Includes a built-in SQLite data source.
- **Lasso Connector for Apache** – Allows Lasso solutions to run via the built-in Apache Web server in Mac OS X.
- **Lasso Connector for IIS** – Allows Lasso solutions to run via the built-in IIS Web server in Windows.
- **Lasso Connector for MySQL** – Allows Lasso to access MySQL data sources.
- **Lasso Connector for FileMaker Pro** – Allows Lasso to access FileMaker Pro 4.x, 5.x, and 6.x data sources.
- **Lasso Connector for FileMaker Server Advanced** – Allows Lasso to access FileMaker Server Advanced 7 data sources.
- **Lasso Connector for JDBC** – Allows Lasso to access JDBC-compliant data sources, including Microsoft SQL Server, PostgreSQL, Frontbase, Oracle, and others.

For complete documentation of the Lasso Professional 8.5 architecture and features, see the *Lasso Professional 8.5* chapter in the Lasso Setup Guide.

Lasso Professional Developer 8

Lasso Professional Developer is a single-user edition of Lasso Professional Server. It has identical features except for two limitations. Only two IP addresses can view Web sites served by Lasso Professional Developer and only 500 transactions will be served per minute. For compatibility testing, Lasso Professional Developer includes a license to run the software on each of the platforms that Lasso supports (Mac OS X, Windows, and Linux).

Lasso Studio for Eclipse

Lasso Studio for Eclipse is a development tool which installs into the Eclipse integrated development environment and provides advanced tools for creating and debugging Lasso pages. Lasso Studio for Eclipse includes an integrated debugger, advanced editing and outlining tools, and more. Solutions built using Lasso Studio are deployed using the full, multi-user version of Lasso Professional Server 8.

Lasso Studio for Adobe GoLive CS
Lasso Studio for Dreamweaver MX

Lasso Studio is an optional development tool for visually building data-driven Web sites within either Adobe GoLive or Adobe Dreamweaver. Lasso Studio consists of a single-user version of Lasso Professional Developer, plus a set of extensions for either Adobe Dreamweaver MX or Adobe GoLive CS. Solutions built using Lasso Studio are deployed using the full, multi-user version of Lasso Professional Server 8.

Documentation

This chapter contains information about the documentation included with Lasso Studio, and describes recommended usage. Comments, suggestions, or corrections to the documentation are appreciated and may be sent to the following email address.

documentation@lassosoft.com

Lasso Studio Documentation

The documentation for Lasso Studio is divided into several different references. All documentation, with the exception of the Lasso Reference, is included in PDF format in the Documentation folder inside the Lasso Studio for Eclipse application folder installed to your hard drive. The following manuals and resources are included.

- **Lasso Studio User Guide** – This is the book that you are reading. It includes documentation for each of the features of Lasso Studio.

The documentation for Lasso Professional is included in the Lasso Professional application folder.

- **Lasso Setup Guide** – This book includes documentation of the architecture and features of Lasso Professional, the administration interface, and Lasso Security topics.
- **Lasso Language Guide** – The documentation of LassoScript, the language used to access data sources, specify programming logic, and much more.
- **Lasso Reference** – Provides detailed documentation of each tag in Lasso. This is the definitive reference to the language of Lasso. This reference is provided as a LassoApp installed with Lasso Professional, and is also available as an online resource from the LassoSoft Web site.

  http://reference.lassosoft.com/

Lasso Studio User Guide

This is the book you are reading. This book contains the following chapters which detail how to install and use Lasso Studio.

- *Chapter 1: Introduction* introduces Lasso Studio and the Lasso 8.5 product line.
- *Chapter 2: Installing Lasso Studio* provides step-by-step instructions for installing Lasso Studio for Eclipse.
- *Chapter 3: Configuring Lasso Professional 8.5* provides step-by-step instructions for configuring the single-user testing version of Lasso Professional 8.5 included with Lasso Studio.

- *Chapter 4: Using Eclipse* provides an introduction to using the Eclipse IDE and introduces key concepts like the Eclipse workspace and how to create Lasso Projects.
- *Chapter 5: Using Lasso Studio* provides an introduction to using the Eclipse editor to create LassoScript files and check their syntax.
- *Chapter 6: Running LassoScripts* describes how to run Lasso pages within Eclipse. Also includes instructions for setting up a run configuration.
- *Chapter 7: Debugging LassoScripts* describes how to debug Lasso pages including setting breakpoints, stepping through Lasso code, simulating HTTP requests, and more.
- *Appendix A: License Agreement* presents the license agreement for Lasso Studio. The Lasso Studio license agreement must be read and agreed to before installing and using Lasso Studio.
- *Appendix B: Glossary* provides a glossary of key terms used within the Lasso Studio User Guide. If you do not understand the meaning of a word when using this guide, please refer to the glossary.

Documentation Conventions

This documentation uses several conventions in order to make finding information easier.

Definitions are indicated using a bold, sans-serif type face for the defined word. This makes it easy to find defined terms within a page. Terms are defined the first time they are used.

Cross References are indicated by an italicized, sans-serif typeface. For instance, the next section in this chapter is *Customer Support*.

Code is formatted in a narrow, sans-serif font. Code includes HTML tags, Lasso tags, and any text which should be typed into a Lasso page. Code is represented within the body text (e.g., [Field] or <body>), or is specified in its own section of text as follows:

```
[Field: 'Company']
```

Code Results represent the results after code is processed. They are indicated by a black arrow, and will usually be the value that is sent to the client's Web browser. The following text could be the result of the code example above.

```
➜ LassoSoft
```

Lasso Tags always appear in code format exactly as they appear except when referring to a group of related tags. For example, -Encoding... refers to all of the encoding keywords (-EncodeNone, -EncodeHTML, etc.). Also, member tags are always referred to beginning with their data type followed by the member tag symbol. For example, the member tag for specifying a character in a text string is referred to as [String->Get].

File Paths can be local or fully-qualified file paths on either Windows or Mac OS X, and are formatted using the same font as code. File paths are represented within the body text (e.g., Lasso Professional 8.5/Admin/Setup or C:\Inetpub\wwwroot), or are specified in their own section of text as follows:

```
C:\Program Files\OmniPilot Software\Lasso Professional 8.5
```

File paths in Mac OS X contain forward slashes (/), while file paths for Windows contain backward slashes (\). If a file path is identical on both Mac OS X and Windows, then the file path will be shown once with forward slashes (/). Users of the command line utility in Windows will need to enter backward slashes instead of forward slashes for these examples.

Note: Notes are included to call attention to items that are of particular importance or to include comments that may be of interest to select readers. Notes may begin with ***Warning***, ***Tip***, ***FileMaker Pro Tip***, ***IIS Tip***, etc. to specify the importance and audience of the note.

To perform a specific task:

The documentation assumes a task-based approach. The contents following a task heading will provide step-by-step instructions for the specific task.

Customer Support

There are many resources available to help you when installing and using Lasso Studio. It is recommended that the following resources be used in the order in which they are presented below.

To find answers to questions about Lasso Studio:

1 **Documentation** – The documentation should be your first resource. Check to see if the Lasso Studio User Guide, Lasso Setup Guide, Lasso Language Guide, or Lasso Reference have the answer to your questions.

2 **LassoSoft Support Central** – A Web resource that allows you to search for updates to the documentation, software updates, and tips about how to best utilize Lasso products.

http://support.lassosoft.com

3 **Lasso Studio Talk Email Discussion List** – A broad community of Lasso Studio users who help to answer each other's questions about using Lasso Studio products. If you can't find an answer to your question, then posting to the list will often result in an answer in a short period of time. Information about subscribing to the list and searchable list archives can be found at the following address.

http://listsearch.lassosoft.com/LassoStudio/

4 **Email and Phone Support** – Provided by LassoSoft to qualified customers during normal business hours. Visit the following URL for more information.

http://support.lassosoft.com/

Usage Rights

These are the usage rights for Lasso Studio. Please consult the Lasso Setup Guide for important information about the usage rights and license agreements specific to those products.

New Purchase

Your license permits a single copy of Lasso Studio to be installed and used on a single computer. While certain components of Lasso Studio (i.e. the single-user testing version of Lasso Professional Developer) may be installed on a separate computer from Eclipse, only a single instance of each component is permitted. The license does not permit development or deployment using non-purchased versions or evaluation versions.

Upgrade Purchase

It is standard industry practice and understood that by upgrading one's software one no longer uses the old version and the license to use and transfer said license ceases.

If you have upgraded to this version of Lasso Studio from any previous version of Lasso, you must no longer use that version. Please see the termination provisions in the accompanying license agreement for further details. Alternatively, you could purchase a new Lasso product license and not be bound by such upgrade restrictions.

Evaluation Versions

Evaluation versions are provided for one-time 30-day evaluation and initial product testing use. Evaluation versions are not licensed for use for extended development. The documentation provided with evaluation versions is to be used strictly within the evaluation time period.

2

Chapter 2
Installing Lasso Studio

This chapter provides a step-by-step guide to installing and initializing Lasso Studio for Eclipse and the single-user edition of Lasso Professional Developer included with Lasso Studio.

- *System Requirements* lists the minimum system requirements for Lasso Studio for Eclipse.
- *Installation Instructions* includes step-by-step instructions for installing Lasso Studio on each supported platform.
- *Getting Started* provides instructions for what to do after Lasso Studio is first installed and initialized.
- *Updating Instructions* includes instructions for updating the Lasso Studio components for Eclipse.
- *Installation Contents* lists the files installed with Lasso Studio.
- *Extended Configuration* describes installing and connecting Eclipse to machines running the full multi-user version of Lasso Professional Server.
- *UnInstalling Lasso Studio* includes step-by-step instructions for removing Lasso Studio from your system.

System Requirements

Lasso Studio will run on systems which meet the minimum requirements listed below. These requirements are for both the Lasso Studio Eclipse components and the included single-user version of Lasso Professional Developer. Although Lasso Studio may run on machines which do not meet these requirements, these installations are not officially supported.

Deployment Note: To deploy Lasso solutions created using Lasso Studio to the Web, a Web server running a full multi-user license of Lasso Professional Server is required, and is purchased separately from Lasso Studio. If you are not hosting your own server, you may serve your solutions via a Lasso ISP. LassoSoft maintains a list of Lasso ISPs at http://www.lassohost.com.

Mac OS X

- G5, G4, or G3 Power Macintosh (or compatible) computer.
- 512 MB of RAM. More recommended.
- Standard installation of Mac OS X (10.3 or higher) or Mac OS X Server (10.3 or higher). This should include an HFS+ formatted hard drive with BSD subsystem option. UFS systems are not supported.
- Default Mac OS X installation of Apache Web Server (included with Mac OS X). Required for the server environment.
- X11 for Mac OS X 10.3 must be installed in order for the [Image] tags in Lasso 8.5 to function. For information on obtaining and installing X11, see http://www.apple.com/macosx/x11/.

Windows 2000/2003/XP

- 300 MHz or higher Pentium-compatible CPU.
- 512 MB of RAM. More recommended.
- Microsoft Windows 2000, Windows XP Professional, or Windows 2003 Server.
- Microsoft Internet Information Services (IIS) 5 or higher. IIS 5 is included with Windows 2000 and Windows XP Professional, and can be installed from the Windows installation CD. Required for the server environment.
- Java Virtual Machine compatible with Sun's JRE 1.4 (Java 2). Required for Eclipse and the Java-based components in the server environment.
- Standard installation of ImageMagick, which can be downloaded at:

  http://lassodownload.blueworld.com/pub/Lasso8/.

Linux

- 300 MHz or higher Pentium-compatible CPU.
- 512 MB of RAM. More recommended.
- Standard installation of Red Hat Enterprise Linux 4 with network support.
- Red Hat Linux installation of Apache Web Server 2 (included with Red Hat Enterprise Linux 4).
- IBM's Java Runtime Environment (JRE) 1.4.1 for Red Hat Linux is required for Java-based tags and JDBC data sources to function. Installation instructions are included below.
- ImageMagick 6.2.0 or greater is required for image manipulation tags to function. Installation instructions are included below.

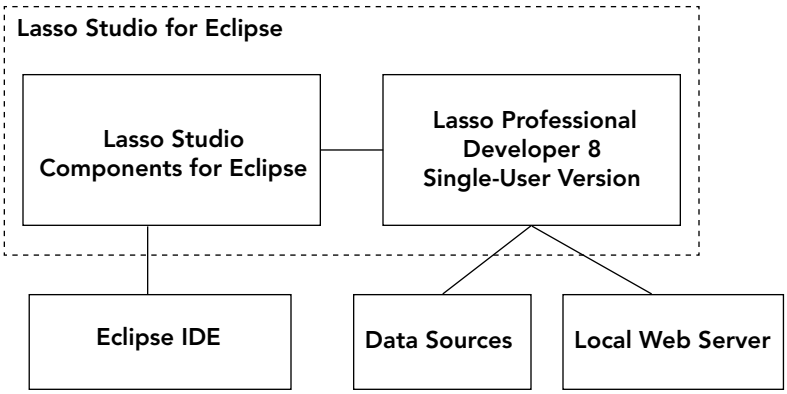
Note: See the included Lasso Professional Server 8 Setup Guide or the Read Me file for details about what packages are required.

Installation Instructions

This section describes installing a default installation of Lasso Studio. A default installation is where both the Lasso Studio components for Eclipse and the included single-user version of Lasso Professional Developer are installed on the same machine. This configuration is recommended for most users.

For best results, please perform the procedures in the following sections in the order they are presented.

Figure 1: Standard Installation



Installing Lasso Professional Developer

Follow this procedure to install the included single-user version of Lasso Professional Developer on your Mac OS X or Windows machine. The version of Lasso Professional Developer included with Lasso Studio for Eclipse automatically installs the debugger module and EclipseSupport.LassoApp files required to run or debug LassoScript files.

To install Lasso Studio for Eclipse on Mac OS X:

1. Double-click the Lasso Studio for Eclipse Installer application.
2. Enter the username and password of a Mac OS X user on the machine who has administrative rights. This launches the Lasso Studio installation window.
3. Select Continue. This will display the license agreement.
4. After reading and agreeing to the terms, select the Accept button. This will display the Lasso Studio Release Notes. The release notes contain important late-breaking information that might not be covered in the documentation.
5. After reading the release notes which describe what is included with Lasso Studio, select Continue.
6. From the pull-down menu, select Easy Install. This will install the included single-user version of Lasso Professional Developer and the debugger module and EclipseSupport.LassoApp files required to use the debugger.
7. Select Install to install Lasso Studio. Mac OS X will prompt for an administrator name and password to perform the installation.
8. Select Quit when the installer has completed. Upon quit the installer automatically restarts the Apache Web server and starts the Lasso Professional server environment.
9. Initialize Lasso Professional Developer with the Lasso Studio serial number or with an evaluation serial number. Configure the server administrator username and password and the site administrator username and password.

Continue to the next section for instructions about how to install the Lasso Studio components for Eclipse from within the Eclipse application itself.

To install Lasso Studio for Eclipse on Windows:

1. Double-click the Lasso Studio for Eclipse Installer.msi application. This launches the Lasso Studio for Eclipse Setup window.
2. Select Next >. This will display the license agreement.
3. After reading and agreeing to the terms, select the I accept the license agreement radio button and select Next >. This will display the Lasso Studio Release Notes. The release notes contain important late-breaking information that might not be covered in the documentation.
4. After reading the release notes which describe what is included with Lasso Studio, select Next >.
5. Select the drive and destination folder to which Lasso Studio will be installed and select Next >. The drive and destination folder is C:\Program Files\OmniPilot Software\ by default. To ensure all features in Lasso Studio to work properly, it is strongly recommended that the \Program Files\OmniPilot Software\ destination folders not be changed.
6. Select the Typical radio button and select Next >. This will install the included single-user version of Lasso Professional Developer and the debugger module and EclipseSupport.LassoApp files required to use the debugger.
7. Select Next > to install Lasso Studio.
8. Select Finish when the installer has completed. Upon quit, the installer automatically restarts IIS and starts the Lasso Professional server environment.
9. Initialize Lasso Professional Developer with the Lasso Studio serial number or with an evaluation serial number. Configure the server administrator username and password and the site administrator username and password.

Continue to the next section for instructions about how to install the Lasso Studio components for Eclipse from within the Eclipse application itself.

To install Lasso Studio for Eclipse for Linux:

1. Copy the Lasso rpm files (e.g. Lasso-Studio-Eclipse-1.5.1-1.i386.rpm, Lasso-Service-8.1.0-1.i386.rpm, LassoDocumentation-8.1.0-1.i386.rpm, Lasso-Apache2Connector-8.1.0-1.i386.rpm) to the Red Hat Linux hard drive into the same folder. These files are unpacked from the LassoStudioforEclipse_Linux.tar.gz archive along with the release notes.
2. Log in as the root user of the Red Hat Linux machine. This is done by entering su in the command prompt, then entering the root password. The hyphen loads the environment for the root user in the resulting shell.

   ```
   su -
   ```

3. Install the Lasso packages. This can be accomplished using the following shortcut or with individual commands for each RPM file.

- Enter the following command to install all three packages for Lasso Professional Server. The rpm command is called with the -i command to perform an install, the -v command for verbose feedback, and -h command to display a progress indicator.

  ```
  rpm -ivh Lasso*.rpm
  ```

- Enter the following commands to install each package for Lasso Professional Server. The rpm command is called with the -i command to perform an install, the -v command for verbose feedback, and -h command to display a progress indicator.

  ```
  rpm -ivh Lasso-Studio-Eclipse-1.5.1-1.i386.rpm
  rpm -ivh Lasso-Service-8.1.0-1.i386.rpm
  rpm -ivh Lasso-Documentation-8.1.0-1.i386.rpm
  rpm -ivh Lasso-Apache2Connector-8.1.0-1.i386.rpm
  ```

4. Initialize Lasso Professional Developer with the Lasso Studio serial number or with an evaluation serial number. Configure the server administrator username and password and the site administrator username and password.

Installing Eclipse Components

The Lasso Studio components for Eclipse are installed using the built-in update feature of Eclipse. This wizard allows Eclipse to discover new features and install them automatically. The following steps should be performed in the order they are presented in order to properly install Lasso Studio into Eclipse.

1. Download and install the Eclipse IDE. The latest supported Eclipse installation can be downloaded from the following Web site.

   ```
   http://www.eclipse.org/downloads/
   ```

   After decompressing the archive Eclipse should be ready to run. The downloaded folder can be moved into the Applications or Program Files directory if desired.
2. Launch the Eclipse IDE by double-clicking on the Eclipse application icon.
3. The Lasso Studio components are installed from within Eclipse. Choose the following menu option to begin the installation.

   ```
   Help > Software Updates > Find and Install...
   ```

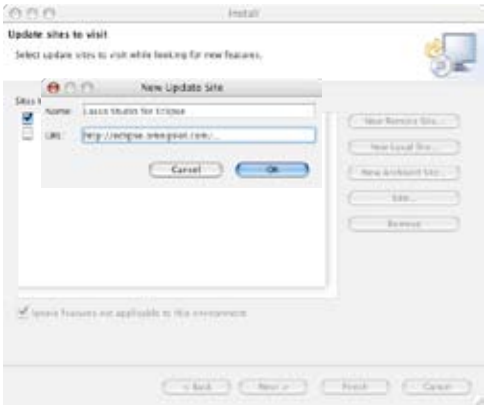
Figure 2: Eclipse Install Screen

Select the Search for new features to install option and click Next >.

4 If Lasso Studio for Eclipse is not present in the list then click on the New Remote Site... button. In the resulting dialog box enter Lasso Studio for Eclipse as the name and the URL http://eclipse.lassosoft.com/.

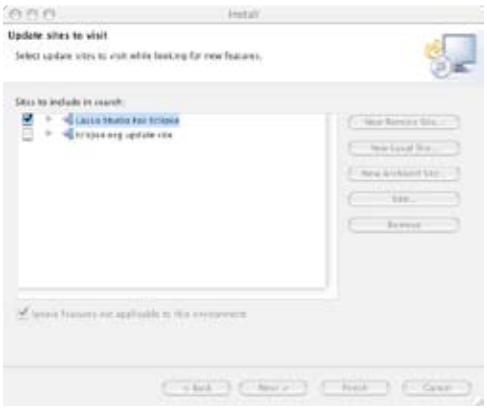
Name - Lasso Studio for Eclipse
URL - http://eclipse.lassosoft.com/

Figure 3: New Remote Site



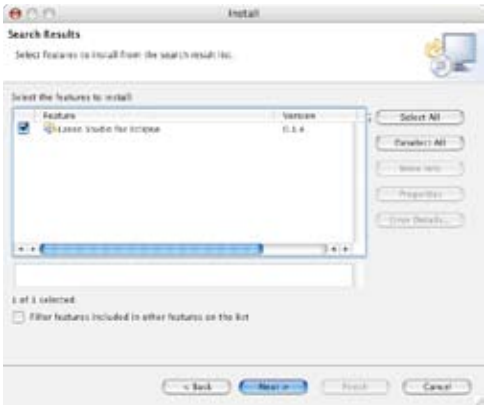
5 The new entry will be added to the Sites to Search listing. Ensure that Lasso Studio for Eclipse is checked and select the Next > button. No other items in the list should be checked.

Figure 4: Sites to Search



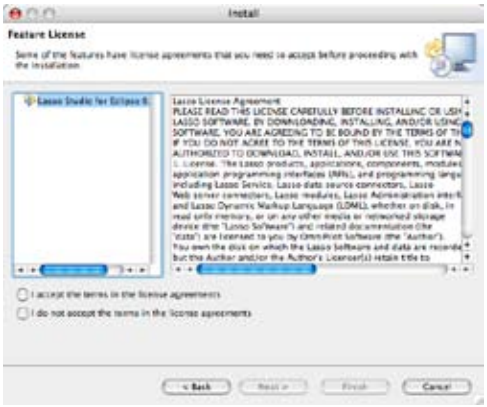
6 Eclipse will search the specified URL and return a list of components that can be installed. The list should include one item Lasso Studio for Eclipse. Make sure that this item is selected and click Next >. No other items in the list should be selected.

Figure 5: Select the Features to Install



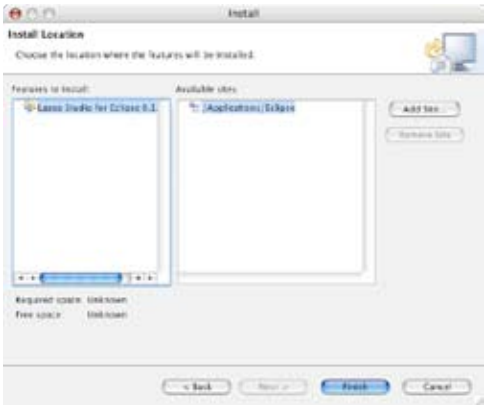
7 The following screen shows the license agreement for Lasso Studio for Eclipse. Read through the license agreement and then select I accept the terms in the license agreements and click Next >.

Figure 6: License Agreement



8 The final screen prompts for the installation location for the Lasso Studio components. The default location should work fine. Select the Next > button to continue.

Figure 7: Install Location



9 Finally, a dialog box will ask whether the Eclipse workspace should be restarted or not. It is necessary to restart the workspace so Yes should be selected.

Figure 8: Restart Workspace?

Initialization

The first time that the Lasso Studio components for Eclipse are accessed after installing Lasso Studio a dialog box will prompt for the Lasso Studio for Eclipse serial number. Enter either a 30-day evaluation serial number or a purchased serial number to continue.

The serial number can also be entered by accessing the Lasso Studio preferences panel within the Eclipse preferences dialog. Select Preferences from the Windows menu and the Lasso Studio > Serial Number panel.

Note: Lasso Studio requires a valid serial number to use any of the debugging features. Many of the editor features such as syntax coloring, code completion, etc. will work without a valid serial number.

Getting Started

The remaining chapters in this user guide provide background information and step-by-step examples for each of the major capabilities of Lasso Studio.

- *Using Eclipse* provides an overview of the Eclipse Integrated Development Environment (IDE) including common terms and interface elements.
- *Using Lasso Studio* introduces the editor-based features of Lasso Studio and the views which report syntax errors, show an outline of container tags, and more.
- *Running LassoScripts* shows how LassoScript files can be run from within Lasso Studio for Eclipse.
- *Debugging LassoScripts* shows how LassoScript files can be interactively debugged within Lasso Studio for Eclipse.

Additional information about Lasso can be found in the Lasso Professional Server 8 Setup Guide and the Lasso Language Guide as well as in the online Lasso Reference.

http://reference.lassosoft.com/

Updating Instructions

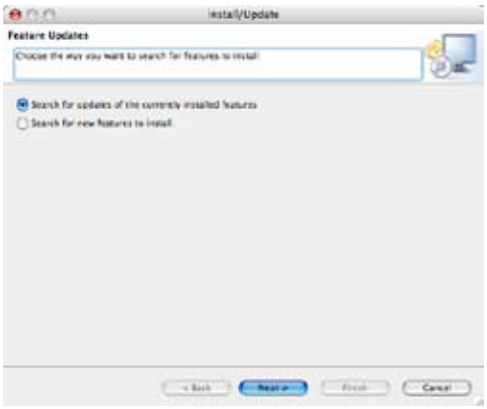
Updates for the Lasso Studio components for Eclipse can be accessed directly from within Eclipse. This section describes how to use the automatic updater within Eclipse to check whether Lasso Studio for Eclipse has been updated. The following steps should be performed in the order they are presented in order to properly update the Lasso Studio components for Eclipse.

Note: If the Lasso Studio components for Eclipse have not yet been installed, follow the instructions in the prior *Installation Instructions* section.

1 Launch the Eclipse IDE and choose the following menu option to check for updates.

Help > Software Updates > Find and Install...

Figure 9: Eclipse Install Screen

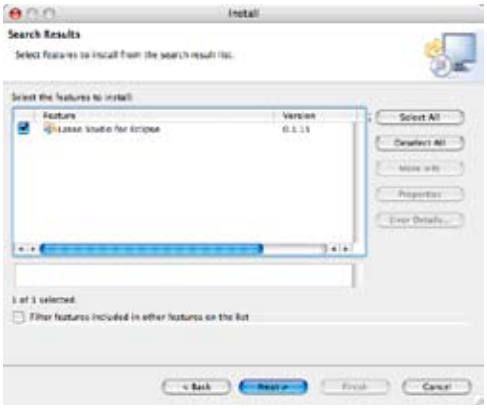


Select the Search for updates of the currently installed features option and click Next >.

2 Eclipse will search and return a list of components that can be updated. The list may contain updates for Lasso Studio for Eclipse or any other features that have been added to Eclipse. If the list does not contain an item for Lasso Studio for Eclipse then no update for Lasso Studio is currently available.

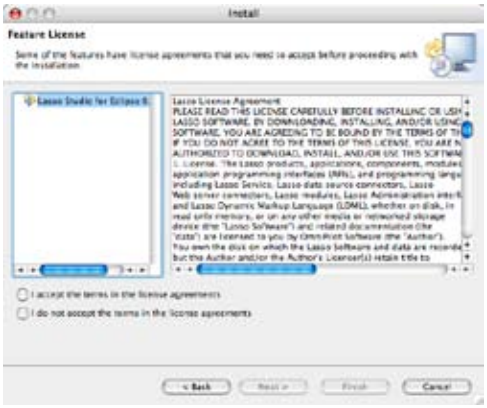
If the list does contain an item for Lasso Studio for Eclipse, make sure that this item is selected and click Next >. No other items in the list should be selected.

Figure 10: Select the Features to Install



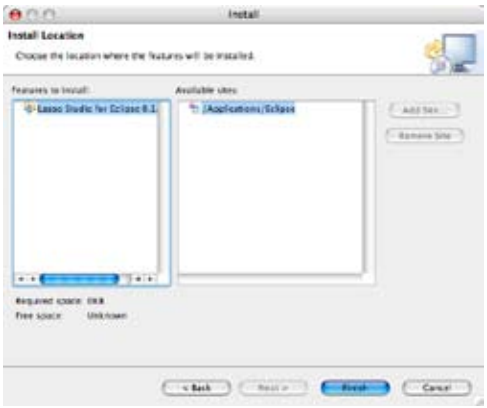
3 The following screen shows the license agreement for Lasso Studio for Eclipse. Read through the license agreement and then select I accept the terms in the license agreements and click Next >.

Figure 11: License Agreement



4 The final screen prompts for the installation location for the Lasso Studio components. The default location should work fine. Select the Next > button to continue.

Figure 12: Install Location



5 Finally, a dialog box will ask whether the Eclipse workspace should be restarted or not. It is necessary to restart the workspace so Yes should be selected.

Figure 13: Restart Workspace?

Note: If any errors are reported after updating the Lasso Studio components for Eclipse, restart the Eclipse application again.

Installation Contents

This section contains a list of all files and folders that are installed during Lasso Studio installation. This includes both the Lasso Studio components for Eclipse and single-user version of Lasso Professional 8.5.

Lasso Studio for Eclipse Files

The following lists all files installed with Lasso Studio. This includes the Lasso Studio for Eclipse folder, and the core files installed within Eclipse.

- **Documentation** – Includes the release notes, change notes, and this user guide.
- **Extensions** – Includes the debugger module and EclipseSupport.LassoApp files which are required in order to connect the debugger in Lasso Studio to a remote installation of Lasso Professional 8.5.
- **LassoAdministration.htm** – A link to the Lasso Site Administration interface on the current machine.

Eclipse Folder

Lasso Studio installs the following files and folders into the Eclipse application folder.

- **Features** – A com.omnipilot.Lasso.LassoStudioforEclipse_# folder is installed which includes a description of the individual plugins that make up the Lasso Studio components for Eclipse. There may be multiple versions of this folder if multiple updates of Lasso Studio have been installed.
- **Plugins** – A com.omnipilot.Lasso.LassoDebuggerPlugin_# folder is installed which includes the debugger components and a com.omnipilot.Lasso.LassoEditorPlugin_# folder is installed which includes the editing components. There may be multiple versions of these folders if multiple updates of Lasso Studio have been installed.

Lasso Professional Files

The following files are installed by the Lasso Studio for Eclipse installer in addition to the files installed with any Lasso Professional installation.

- **LassoModules** – The debugger module is installed in this folder. This module has a different name depending on what platform is being used.

  Debugger_DBGp.dylib
  Debugger_DBGp.so
  Debugger_DBGp.dll

- **LassoStartup** – An EclipseSupport.LassoApp file is installed. This file creates the Lasso Studio group and defines several custom SOAP tags required by Lasso Studio for Eclipse..

For a list of all files installed with the included single-user version of Lasso Professional, see the *Installation Contents* sections in the installation chapters of the Lasso Setup Guide.

The included single-user installation of Lasso Professional Developer is identical to the full version except the core Lasso Service file accepts Lasso Studio serial numbers to go into single-user mode. The single-user version may be converted to a full multi-user version simply by entering a purchased Lasso Professional Server serial number in Lasso Site Administration.

Extended Configuration

This section describes how to use Lasso Studio with an installation of Lasso Professional Developer on a different machine.

- The Lasso Studio components for Eclipse can be downloaded and installed from within Eclipse on one machine and the Lasso Studio for Eclipse installer can be used to install the included edition of Lasso Professional Developer on another machine.
- An existing copy of Lasso Professional Developer can be configured to support remote debugging by installing the debugger module into the LassoModules folder and EclipseSupport.LassoApp into the LassoStartup folder.

Note: It is not recommended that the remote debugger be used with an installation of Lasso Professional Server. The debugger can interfere with the performance of Lasso Professional and the debugger triggers can prevent the normal serving of Web pages while an interactive debugger session is active.

Uninstalling Lasso Studio

This section describes uninstalling both the Lasso Studio components for Eclipse and single-user version of Lasso Professional Developer.

Eclipse Components Uninstallation

The components for Eclipse should be removed from within the Eclipse interface using this procedure.

1. Select the following menu option from within Eclipse Help > Software Updates > Manage Configuration...
2. Find and select the Lasso Studio for Eclipse item in the list of installed components on the left.
3. Choose the Uninstall option from the list of option on the right.

The components for Eclipse can be easily re-installed later using the instructions presented earlier in this chapter.

Lasso Professional Developer Uninstallation

- **Mac OS X** – The download package includes an uninstaller that can be used to remove the included Lasso Professional Developer edition. See the section on *Files and Folders Not Removed* below for details of what files will remain on the machine.

  Uninstall Lasso Studio for Eclipse.mpkg

- **Windows** – Use the Add/Change Programs control panel to remove the Lasso Studio for Eclipse package. See the section on *Files and Folders Not Removed* below for details of what files will remain on the machine.
- **Linux** – Log in as the root user of the Red Hat Linux machine. This is done by entering su in the command prompt, then entering the root password. Stop Lasso Service using the lasso8ctl command as shown below. Uninstall the Lasso packages. This can be accomplished using the following shortcut command to remove all Lasso packages or by removing the three Lasso packages individually.

  Enter the following commands to uninstall all three packages for Lasso Professional Server. This command lists all installed rpm files, filters the list for those that contain Lasso, and then pipes the list to the rpm program with the remove option -e.

  ```
  su -

  /usr/sbin/lasso8ctl stop

  rpm -qa | grep Lasso | xargs rpm -e
  ```

  Alternately, enter the following commands to remove each Lasso package individually. Note that the remove command does not require the version number for each package to be specified.

  ```
  su -

  /usr/sbin/lasso8ctl stop

  rpm -e Lasso-Service
  rpm -e Lasso-Documentation
  rpm -e Lasso-Apache2Connector
  ```

Files and Folders Not Removed

The Lasso Studio installer program will not remove any non-default files created after the time of install, or any default Lasso files modified since the time of install. These files include the SQLite databases, Lasso log files, Lasso setups, and custom Lasso pages and LassoApps. These files remain in the following locations:

- **Lasso Professional 8.5 folder** – The /Applications/Lasso Professional 8.5 or Program Files\OmniPilot Software\Lasso Professional 8.5 folder is left on the hard drive, which contains folders with any custom modules, JDBC drivers, LassoApps, Lasso libraries, sites, and SQLite databases that have been created.:
- **Web Serving folder** – The Web serving folder (e.g. /Library/WebServer/Documents or C:\Inetpub\wwwroot) retains all Lasso pages created or modified since the time of installation. Customized Lasso pages, Web pages, and scripts will not be deleted during uninstallation.

These files and folders may be backed up for preservation, or may be deleted to completely remove Lasso Studio and all settings from the system.

Reinstallation Note: Any files remaining on the system from a previous installation of Lasso Studio will not be overwritten by installing a newer version. Therefore, all previous settings will be retained if a newer version of Lasso Studio is installed over the files not removed during the uninstallation of the previous version.

3

Chapter 3
Configuring Lasso Professional 8.5

This chapter contains instructions for configuring a Lasso Professional server environment for use with Lasso Studio.

- *Using Lasso Professional 8.5* describes how to use Lasso Professional 8.5 in relation to Lasso Studio.
- *User Configuration* describes setting up custom users for use with Lasso Studio instead of the Lasso site administrator account. This section is required to use the debugger with any user other than the site administrator and is recommended for ISPs and advanced administrators.

Using Lasso Professional 8.5

This section describes how to operate your Lasso Professional server environment so that it may be used with Lasso Studio. For a full description of the features, usage, and architecture of Lasso Professional, see the *Lasso Professional 8.5* chapter in the Lasso Setup Guide.

Running Lasso Professional 8.5

Lasso Professional consists of one main service known as Lasso Service and one or more child processes. Lasso Service is the core executable of Lasso Professional which communicates with a Web server. The sections below describe starting Lasso Service on Mac OS X, Linux, and Windows.

Mac OS X

Lasso Service is started and stopped using the terminal or scripts located in the /Applications/Lasso Professional 8.5/Tools folder.

- Type the following command into the Mac OS X Terminal application. You will be prompted for your password. This command must be accessed from an administrator account.

```
sudo lasso8ctl start
```

- Alternately, open the Tools folder in the Lasso Professional Server application folder and double click on the startLassoService.command file. This will launch a terminal window that prompts you for your password and starts Lasso Service.

For complete instructions on how to start and stop Lasso Service on Mac OS X using these scripts, see the *Configuring on Mac OS X* chapter *Running Lasso Professional 8.5* section in the Lasso Setup Guide. This section also provides instructions for how to check if Lasso Service is running and how to run Lasso in console mode.

Windows

Lasso Service is started and stopped using the Services control panel.

- From the Windows Task Bar, select *Start > Settings > Control Panel > Administrative Tools > Services*. Right click on Lasso Professional Server 8, then select Start to start Lasso Service, or Stop to stop Lasso Service.

For complete instructions on how to start and stop Lasso Service on Windows using the services menu, see the *Configuring on Windows* chapter *Running Lasso Professional 8.5* section in the Lasso Setup Guide. This section also provides instructions for how to check if Lasso Service is running and how to run Lasso in console mode.

Linux

Log in as the root user of the Red Hat Linux machine. This is done by entering su in the command prompt, then entering the root password.

```
su -
```

- Use the lasso8ctl command to start Lasso Service.

```
/usr/sbin/lasso8ctl start
```

- Alternately, cd into the Tools folder and run the startLassoService.sh shell script.

```
cd "/usr/local/Lasso Professional 8.5/Tools"
./startLassoService.sh
```

Lasso Site Administration

Lasso Site Administration is a convenient, Web-based interface for configuring Lasso Professional global settings, configuring data source connections, configuring Lasso Security, monitoring events, and much more. It is within this interface that all Lasso settings and databases are set up and configured.

Figure 1: Lasso Site Administration

Lasso Site Administration consists of the SiteAdmin.LassoApp file served directly from the LassoApps folder where Lasso Professional 8.5 is installed. For more information on LassoApps, see the *LassoApps* chapter in the Lasso Language Guide.

Accessing Lasso Site Administration

In a Web browser, visit http://www.example.com/SiteAdmin.LassoApp. Replace www.example.com with your domain name or IP address of the Web server running Lasso Professional 8.5. Use 127.0.0.1 as the IP address if on the same machine. If an error is displayed, make sure Lasso Service is running on the Web server as described previously.

Using Lasso Site Administration

An overview of the sections and features of Lasso Site Administration is in the *Using Lasso Site Administration* chapter of the Lasso Setup Guide. For instructions on how to set up users other than the Lasso site administrator for use with Lasso Studio, see the *User Configuration* section of this chapter. This is recommended for ISPs and advanced administrators.

Creating Lasso Studio Users

Users which are added to the Lasso Studio for Eclipse group will have the ability to check and execute code through Lasso Studio.

To add a user to the Lasso Studio for Eclipse group:

1. In Lasso Site Administration, go to *Setup* > *Security* > *Users*.
2. In the User Listing panel, select the Add User button. This displays the Add User panel to the right.
3. Enter a username in the Username field. This will be used within Eclipse to connect to the server environment.
4. Enter a password in the Password field. This will be used within Eclipse to connect to the server environment.
5. Select the name of the Lasso Studio for Eclipse group in the Groups field to which the user is to be assigned.
6. Select Add User.

Figure 2: User Listing

Note: If the Lasso Studio for Eclipse group is not defined make sure that the EclipseSupport.LassoApp and the debugger plug-in are installed in Lasso Professional.

4

Chapter 4
Using Eclipse

This chapter introduces the Eclipse Integrated Development Environment (IDE).

- *Eclipse Application* describes the Eclipse application and where new versions can be downloaded.
- *Eclipse Workbench* describes the user interface of Eclipse.
- *Eclipse Projects* describes how Eclipse manages files.
- *Lasso Projects* describes how to create Lasso-specific projects within Eclipse, how to create LassoScript files, and how to import and export Lasso files from Eclipse.

Eclipse Application

Eclipse is a Java-based application that runs on Windows, Mac OS X, and Linux. The user-interface of Eclipse is largely the same no matter what platform it is running on.

Eclipse can be downloaded from the Eclipse Project Web site at the following URL. Follow the Platform: Downloads link in the Eclipse Platform section and download the Latest Release.

http://www.eclipse.org/downloads/

Note: The Eclipse application may have a different name in the dock or start bar once it is running. For example, on Mac OS X Eclipse may show up in the dock as java_swt, but with the Eclipse icon.

Eclipse Workbench

Eclipse uses a single window with a number of different parts for its interface. The parts of this main window are described here.

- **Workbench** – The main window of Eclipse is called the workbench. All of the views and editors that present information and allow files to be modified are displayed as parts of the workbench.
- **Perspective** – The current arrangement of views and editors in the workbench is called the current perspective. Perspectives can be saved and recalled to switch between several different custom configurations.

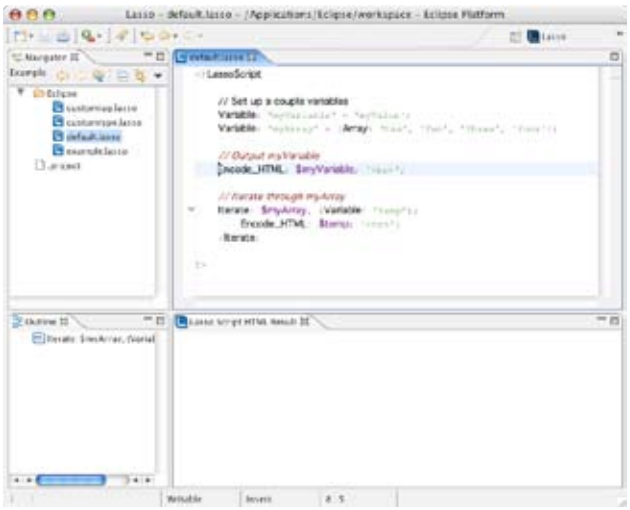
  The pop-up in the upper left corner of the window shows the current perspective and allows a different perspective to be shown. Alternately, the Window > Open Perspective menu can be used to select a perspective. Use the Other... option if the desired perspective is not in the sub-menu.

  There are two perspectives that are of particular use with Lasso Studio:

  Lasso – This perspective allows LassoScripts to be edited, provides information about syntax errors, and shows the results of running LassoScripts. This perspective is covered in detail in the *Using Lasso Studio* and *Running LassoScripts* chapters.

Debug – This perspective allows LassoScripts to be debugged. It will be switched to automatically when a debugging session is started. This perspective is covered in detail in the *Debugging LassoScripts* chapter.

Figure 1: Lasso Perspective



- **View** – A view is a part of the workbench that displays information. Some views display file listings or an outline view of a code files. Other views display syntax errors, runtime errors from the debugger, or the results of executing a LassoScript. Some views allow information to be changed and any changes are saved automatically.

  Some common views include:

  Navigation – Displays the files that are in the current project in an outline view. Code files can be opened by double clicking on their name in this view. Or, they can be run or debugged by selecting the appropriate option from the context menu for the file. Located in the upper-left of the workspace by default.

  Outline – Displays an outline of the container tags for the file in the active editor. The outline can be used to quickly see the structure of a Lasso file and to navigate through the file. Located in the lower-left of the workspace by default.

  Problems – Displays any syntax errors that are found in the file in the active editor. Located below the editor by default.

  LassoScript HTML Result – Displays the result of running a file. This view is described in detail in the *Running LassoScripts* chapter.

  Debugger – Displays feedback from the debugger. This view is described in detail in the *Debugging LassoScripts* chapter.

- **Editor** - A text editor that displays a code file. Editors require that files be saved explicitly. Editors will display Lasso code with syntax coloring and disclosure triangles so portions of the code can be collapsed. Multiple editors can be open at once with tabs to select between them.

The edges of most editors and views can be dragged to change the relative size of the part. In addition, the parts can be moved to different positions relative to one another. The tab of an editor can be dragged toward the bottom of the editor window in order to split the window between two files.

Note: If the workbench gets messed up the Window > Reset Perspective command can be selected in order to restore the default perspective.

Eclipse Projects

Eclipse handles files and folders in projects. A project is a collection of files and folders that are managed by Eclipse. Eclipse has a great deal of flexibility for file management. Files can either be edited directly within the file system or can be imported into Eclipse and stored internally. Eclipse can also work with version control systems such as CVS.

Local Projects

A local project manipulates files directly in the file system. For example, the files in the Web server root can be modified directly without performing any imports or exports.

Managed Projects

A managed project imports files into Eclipse. The files can be modified within Eclipse and must be exported before the changes are reflected in the original files. Managed projects may be used when the files are stored on a remote server. A common Eclipse work flow is as follows:

1. An existing set of Lasso pages are imported into a new Eclipse project. These files are copied into Eclipse Any changes made within Eclipse are not automatically reflected in the original files.
2. Modifications are made to the files using the editors within Eclipse.
3. The files are exported back into the file system (e.g. into the Web server root).

Steps 2 and 3 can be repeated as many times as necessary. New files can be created directly within Eclipse as part of the project and additional files can be imported if necessary.

Lasso Projects

In order to use the Lasso-specific tools in Eclipse it is necessary to create a Lasso Project. This can be accomplished by selecting the *New > Lasso Project* menu command or by selecting *New > Project...* and then choosing to create a Lasso Project in the New Project wizard.

A new Lasso project has two attributes. The project name controls how the project will be labeled in the navigator. The project contents controls where the project will be stored in the file system. By default this location is within the Eclipse application folder, but it can also be set to any location within the file system by unchecking the Use Default checkbox.

All of the projects that have been created are shown in the navigator. In order to view the files from just one project the context menu on a project can be used to Go Into that project. The controls in the navigator toolbar can then be used to return to the default view.

Note: The remaining chapters in this book assume that a Lasso Project has been created. If at any point the menu options and views described in this user guide are not present in Eclipse make sure that a Lasso Project is selected in the navigator view.

Examples

This section contains four examples that demonstrate how to perform common tasks with Lasso Projects. The four examples include:

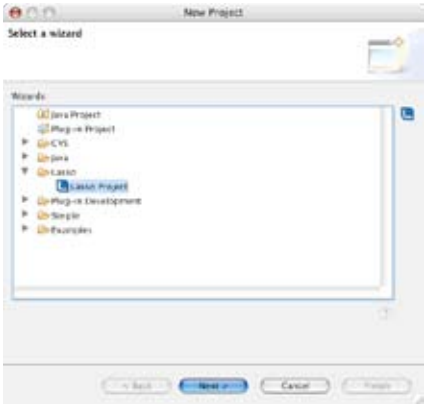
- Creating a Lasso Project based on the Web server root.
- Creating a new, empty Lasso Project.
- Creating a LassoScript file within a Lasso Project.
- Importing files from the current machines's file system into a Lasso Project. This allows an existing folder of Lasso files to be imported so that they can be worked on within Lasso Studio. The imported files are copied into the Eclipse workspace. Changes to these files are not reflected in the original files until an export is performed.
- Exporting files from the current Lasso Project back into the machine's file system. If a folder was imported into Eclipse it can be exported in order to overwrite the original files with the new edited versions of the files. New Lasso Projects can also be exported into the file system (e.g. into the Web server root) once they are completed and read to be served.

To create a Lasso Project based on the Web server root:

This example shows how to create a Lasso Project which includes all of the files in the Web server root. The files will be modified in-place so no imports or exports are required.

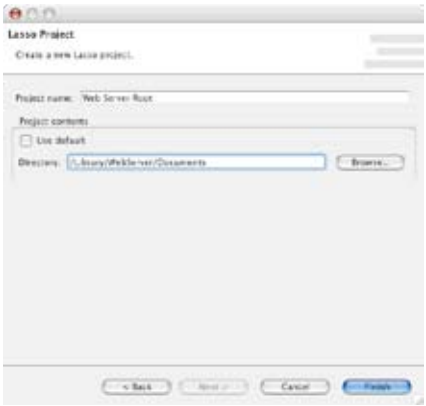
1 Select the menu option File > New > Project.... This will display the new project dialog box shown below.

Figure 2: New Project Dialog



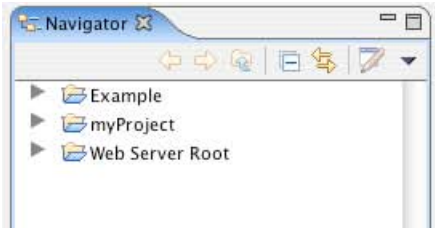
2 Select the Lasso Studio option to create a new Lasso Project and then select Next > to continue.

Figure 3: New Lasso Project Dialog



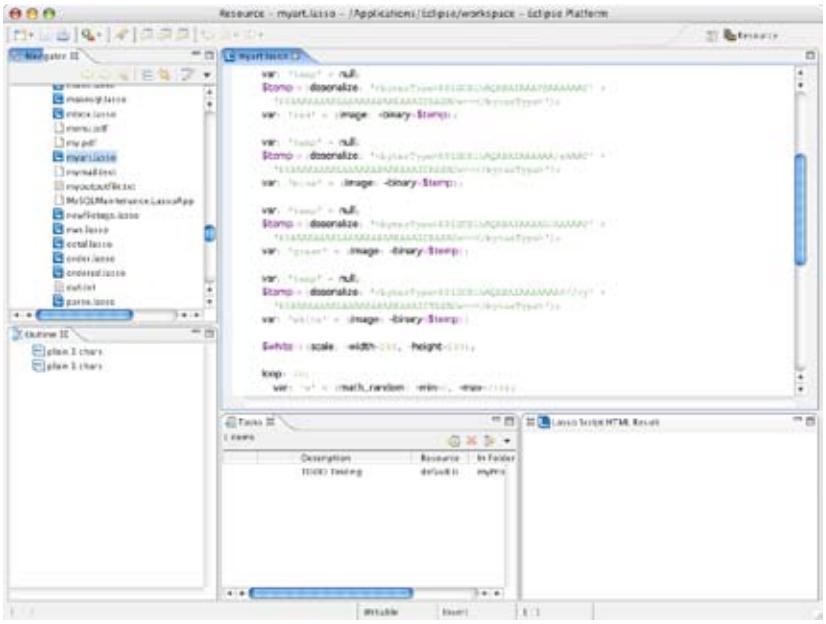
3 The dialog box allows the name for the Lasso Project to be specified. Web Server Root in this example. The Use Default checkbox should be unchecked and then the Browse... button used to select the location of the Web server root. The Mac OS X root for the built-in Apache Web server /Library/WebServer/Documents is shown above. On Windows the root would typically be C:\InetPub\wwwroot\. Other Web servers might have different roots.

Figure 4: Lasso Project Navigator



4 The new project is shown in the navigator view. The contextual menu on the Web Server Root item can be used to Go Into the project so only its files are shown. The following screen shot shows how the entire workspace looks with a LassoScript file shown in the editor..

Figure 5: Lasso Project Editor

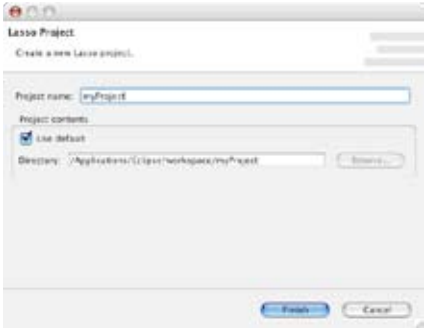


To create a new, empty Lasso Project:

This example shows how to create a new, empty Lasso Project. The following examples show how to create a new LassoScript file and how to import files the from the Web server root into the project.

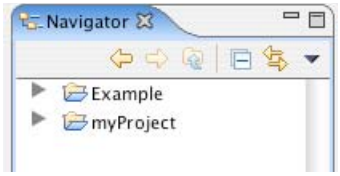
1 Select the menu option File > New > Lasso Project (or choose File > New Project... and select Lasso Project). This will open the dialog box shown in the figure below. Type the name of the new project, myProject for this example, and select the Finish button.

Figure 6: New Lasso Project Dialog



2 A new project is created and can be seen within the navigator view. The figure below shows two projects Example and the new MyProject. If the navigator does not show the new project then the up-folder button may need to be selected in order to return to the top of the projects hierarchy.

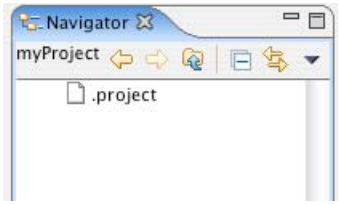
Figure 7: Navigator View



3 Since the navigator can contain many projects it is often desirable to view the files just for one particular project. In the contextual menu for the myProject folder select Go Into. The up-folder button can be used to return to the original view.

The files within myProject are shown in the figure below. The .project file contains project settings and can be ignored. This project is essentially empty.

Figure 8: Navigator View



To create a new LassoScript file:

This example shows how to create a new LassoScript file in a project. This technique can be used in any Lasso Project to add an additional file.

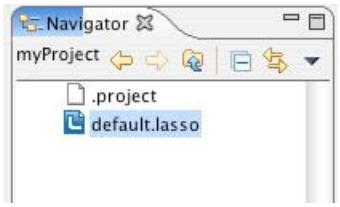
1 Create a new Lasso file in the project by selecting the menu option File > New > LassoScript File. The dialog box shown in the figure below allows the name and project of the new file to be specified. For this example a file named default.lasso is created in myProject. Selecting the Finish button creates the new file.

Figure 9: New LassoScript File Dialog

2 A dialog will ask whether the LassoScript File Template should be used for the new file. The file template can be modified in the Lasso Studio preferences.

3 Now the navigator view shows the new file within myProject.

Figure 10: Navigator View



4 Double-clicking on the file name will open the new file in the editor so that it can be modified. New LassoScript files are usually created with the LassoScript file template, but can be filled with HTML, square bracket syntax, etc.

To import existing files into a Lasso Project:

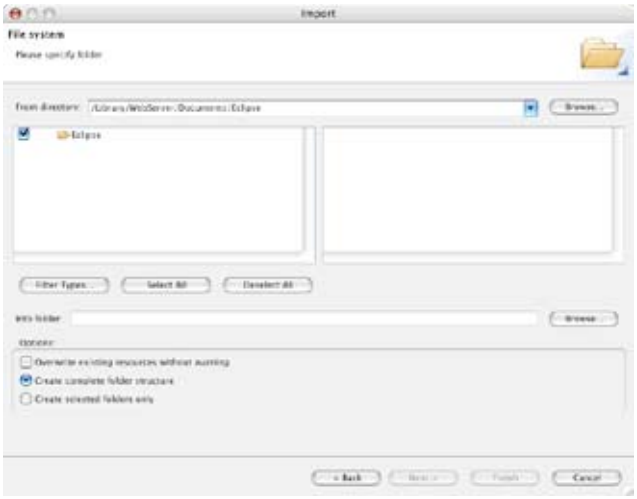
This example shows how to import a folder of existing files into a Lasso Project. This procedure actually copies the files into the Eclipse workspace. The original files will not be modified until the project is exported again.

1 Select Import... from the file menu. This will bring up the dialog box shown in the figure below. Select File System to import files from a folder on the current machine. Select Next > to continue.

Figure 11: Import Select Dialog Box

2 The resulting dialog box is shown in the following figure. The required steps to complete this dialog box are described after the figure.

Figure 12: Import File System Dialog Box



First, use the upper Browse... option to select the desired file from the current machine's hard disk. For this example a file in the Mac OS X Web server root is selected.

/Library/WebServer/Documents/Eclipse

Next, make sure that the checkbox next to the selected folder is checked. In this case the Eclipse folder needs to be checked for it to be imported.

Finally, choose the import location using the lower Browse... option. This will bring up a dialog box which allows a location within the current workspace to be selected. For this example, myProject is selected.

Figure 13: Import Select Dialog Box

3 The files in the selected location are imported into the Eclipse workspace and show up in the navigation view. Notice that the entire hierarchy of the file system down to the location of the files is inserted in the navigator. The Go Into option from the contextual menu and the up-folder control can be used to navigate through the various imported folders.

Note: These files are now under the control of Eclipse. Any changes made to these files will not be reflected in the original folder until the files are exported from Eclipse. See the following example for details.

Figure 14: Import Select Dialog Box

To export files from Eclipse:

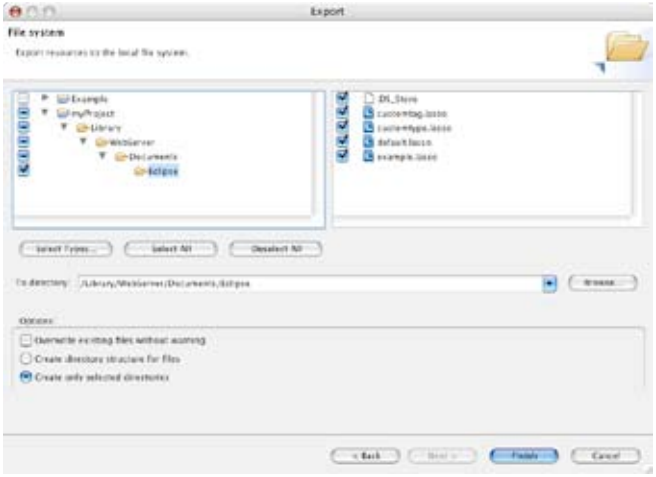
In order for changes to files that are under the control of Eclipse it is necessary to export those files back into the file system. This allows sites to be worked on within the Eclipse environment and then exported for general serving once they are completed.

1 Select the Export... option from the file menu. The dialog box shown in the following figure allows the type of export to be selected. Choose File System and select Next > to continue.

Figure 15: Export Select Dialog Box

2 The resulting dialog box is shown in the following figure. The required steps to complete this dialog box are described after the figure.

Figure 16: Export File System Dialog Box



First, select the files to be exported. For the example above the disclosure triangles in the folder listing were used to expose the Eclipse folder. Selecting the checkbox next to this folder automatically selects each of the files contained within.

Next, use the Browse... button to select where in the file system the export should occur. The same folder from which these files were imported is selected.

/Library/WebServer/Documents/Eclipse

Select the Finish button to perform the export.

3 The export is going to overwrite the files that are already located in the selected folder. A dialog box confirms that the files should be overwritten. Selecting Yes To All will allow the export to occur without any further dialog boxes.

Figure 17: Export File System Dialog Box

5

Chapter 5
Using Lasso Studio

This chapter provides an overview of using Lasso Studio to build Lasso solutions using the Eclipse IDE.

- *Overview* provides an overview of how to use Lasso Studio and the interface elements provided by Eclipse.
- *Code Editing* describes features that make it easier to edit LassoScript code within Lasso Studio including syntax coloring, code folding, outline selections, custom tag and custom data type templates, automatic tag completion, built-in Lasso Reference, and variable completion.
- *Syntax Checking* describes how Lasso Studio automatically syntax checks documents when they are saved and provides a list of errors directly within Eclipse.
- *Keyboard Shortcuts* discusses the keyboard shortcuts that are available in Lasso Studio and how to customize them.
- *Preferences* describes the preference panes that control how Lasso Studio formats code, allow code templates to be created, control what tags show up in the outline view, and more.
- *File Associations* discusses how to associate additional file types with the Lasso Studio editor.

Note: See the following chapter on ***Debugging*** for information about checking LassoScripts for runtime errors and for information about using Eclipse's interactive debugger on LassoScripts.

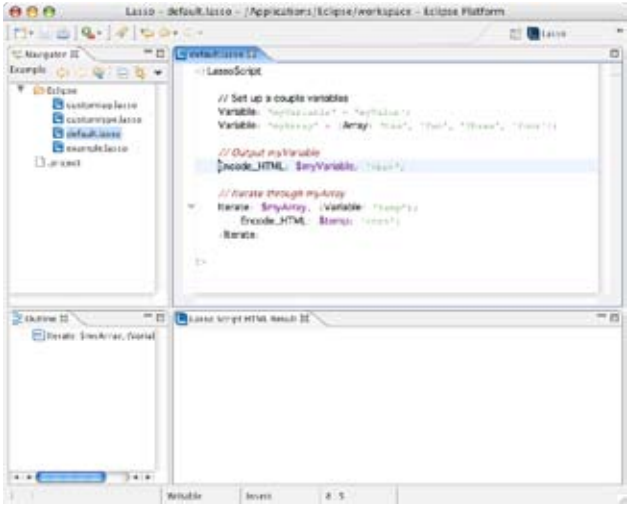
Overview

Lasso Studio is integrated into the Eclipse IDE and many of its features are accessed identically to how those features are accessed when Eclipse is being used to create other types of code. Understanding the basic interface elements of Eclipse is essential to getting the most of out of Lasso Studio.

Interface Elements

The following figure shows the basic interface elements of the Lasso perspective within Eclipse. This perspective defines a standard set of views that allow LassoScripts to be edited and for syntax errors to be reported.

Figure 1: Interface Elements



Note: If the interface of Eclipse does not appear as above then select the Lasso option (with the blue L logo) from the list of perspectives in the upper-right portion of the window. If Lasso does not appear in the option then select Open Pespective > Other... from the Window menu and choose Lasso from the list.

The interface is split into four views and editors which are described below.

- **Navigator** – The navigator shows the files that are open in the current project. A file can be opened in the editor by double-clicking on its name.
- **Editor** – The editor shows one or more open files. A different file can be selected by choosing a different tab.
- **Outline** – The outline shows a hierarchical view of the container tags within the current file shown in the editor. A given container tag can be selected by double-clicking on its name in the outline. The contextual menu in the outline view can be used to insert custom tag or custom data type templates.
- **LassoScript HTML Result / Problems** – The bottom of the window contains two views. The LassoScript HTML Result view displays the result of running the page (more in the next chapter). The Problems view displays any syntax errors that are reported by Lasso Studio.

These interface elements can be moved relative to one another and saved as a new perspective. This default arrangement can be restored by reselecting the Lasso perspective.

Code Editing

With Lasso Studio installed Eclipse supports a number of features that make editing Lasso code easy. These include

- **Syntax Coloring** – Lasso Studio colors different language elements for ease of reading and editing LassoScripts.
- **Tag Outline** – The outline view allows the hierarchical structure of a file to be seen and for different container tags to be selected.
- **Outline Markers** – Comments that start with MARK will be shown in the outline view so they can be easily selected.
- **Task Comments** – Comments that start with TODO will be shown in the tasks view. This allows a list of to do items to be maintained.
- **Tag and Data Type Templates** – The contextual menu within the outline view allows custom tag or custom data type templates to be inserted into the current file.

- **Code Folding** – Container tags can be temporarily folded closed in order to hide their contents and allow the surrounding tags to be read easier.
- **Tag Completion** – Typing ctrl-space while typing in the editor brings up a list of possible tag names for the current insertion point.
- **Templates** – Custom templates allow code to be inserted with placeholders for database names, table names, and other values.
- **Lasso Reference** – The tag completion menu additionally shows the Lasso Reference entry for the current selected tag.
- **Variable Completion** – Typing a $ or # character will bring up variable name recommendations.

Each of these features is described in more detail below.

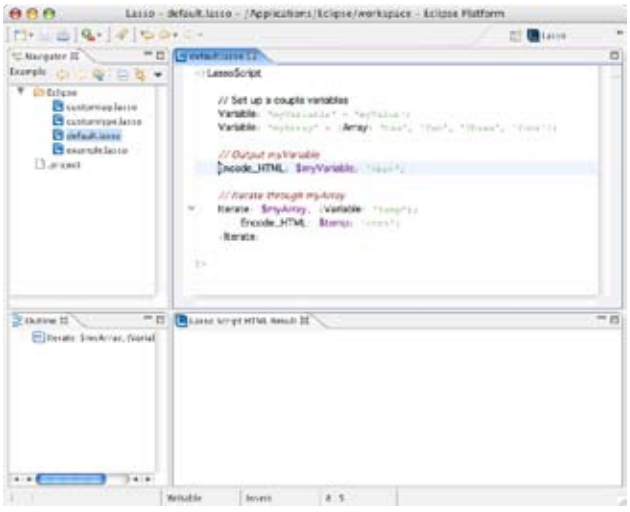
Syntax Coloring

LassoScript consists of a number of different elements including tag names, keywords, literals, variable names, symbols, and more. Lasso Studio for Eclipse colors each of these language elements differently in the editor pane. This makes LassoScript code easier to read and makes it easier to spot certain kinds of syntax errors. Lasso Studio colors the following language elements:

- **Tags, Types, Constants** – Tag names like [Encode_HTML] or [Variable], type names like [Email_POP] or [Array], and constant names like [Net_TypeTCP].
- **Keywords** – Tag parameters that begin with a hyphen - like -Search, -Database, -Table, -Find, -Replace, -EncodeHTML, etc.
- **String Literals** – Strings enclosed within single quotes 'string' or double quotes "string".
- **Numeric Literals** – Integer values like 3 or -1500 and decimal values like 123.456.
- **Operators** – Symbols like +, -, *, /, %.
- **Page Variables** – Page variable references that start with a dollar sign $ like $myVariable or $myArray.
- **Local Variables** – Local variable references that start with a hash mark # like #myLocal.
- **Comments** – Single line comments that begin with a double slash // or multi-line comments delineated by C-style markers /* ... */.

The following figure shows a LassoScript that has syntax coloring applied.

Figure 2: Syntax Coloring



The colors that Lasso Studio uses for the different language elements can be modified by selecting the Preferences... option from the Eclipse menu and then choosing Lasso Studio from the list at left. See the *Preferences* section in this chapter for details of how to change the syntax coloring preferences.

The figure below shows the available options including the color and text style for various language elements and color and font settings for the editor window itself.

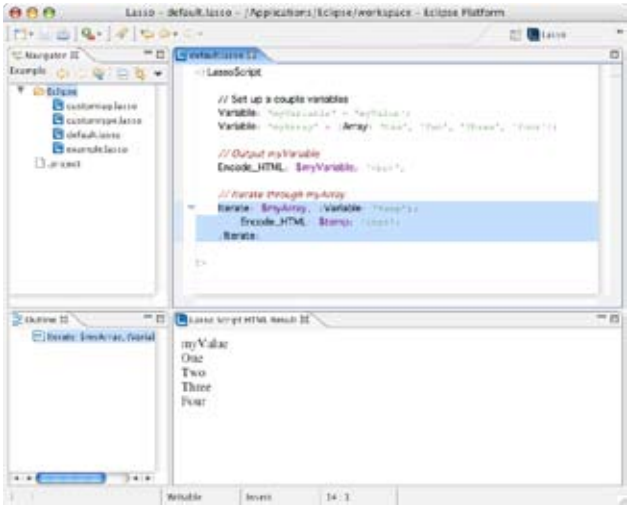
Tag Outline

The tag outline which is in the lower-left corner of the window by default can be used to select any container tags within the LassoScript. This makes it easy to navigate through a complex document by selecting the container tags that create the logical structure of the file.

The figure below shows the result of double-clicking on the Iterate element in the outline. The resulting selection makes it easy to see the extent of the [Iterate] ... [/Iterate] tags.

The tags which are included in the tag outline can be modified in the preferences for Lasso Studio. Select Preferences from the Windows menu and then the Lasso Studio > Ouline View pane to modify which container tags and other elements are included in the outline view. See the *Preferences* section in this chapter for more details.

Figure 3: Outline Selection



Outline Markers

Comments are not normally shown in the outline view, but any comment that begins with MARK will be. This allows markers to be inserted into LassoScript files which can be quickly jumped to using the outline.

```
// MARK This is a marker.
```

To Do Tasks

A comment that beings with TODO will be included in the standard asks view within Eclipse. The tasks view can be shown by selecting the menu command Window > Show View > Tasks. Double clicking on a task will open the file that contains the to do comment.

```
// TODO This is a task.
```

Tag and Type Templates

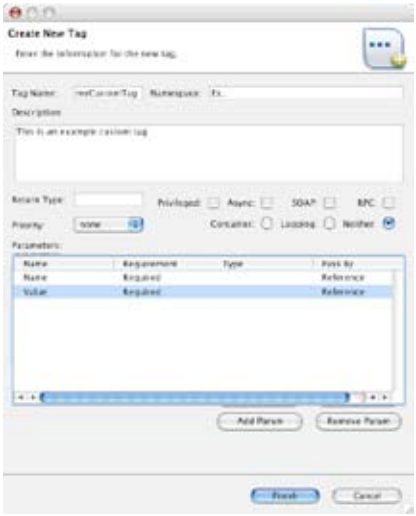
The contextual menu within the outline view allows custom tag or custom data type templates to be inserted into the current file being edited. To use the templates select either Tag... or Type... from the context menu. A dialog box allows the attributes of the new custom tag or custom data type to be selected and then the template is inserted at the current cursor location.

To insert a custom tag template:

Select Tag... from the contextual menu in the outline view. The dialog box shown in the figure below allows the attributes of the custom tag to be specified. These include:

- **Name** – The name of the custom tag. In this example myCustomTag is specified.
- **Namespace** – The namespace of the custom tag. In this example specifying Ex_ means the tag will be callable as [Ex_myCustomTag].
- **Description** – An optional description for the tag.
- **Options** – Many options for the tag can be specified including -Priority, -Privileged, -ReturnType, -Async, -SOAP, -RPC, etc.
- **Parameters** – Both -Required and -Optional parameters can be specified including -Type and -Copy options. Add parameters by choosing Add Param and then edit them directly within the listing.

Figure 4: Custom Tag Template



After selecting Finish the tag is inserted into the current file. The dialog above would create the following code (reformatted for easier reading). The contents of the tag can then be inserted between the [Define_Tag] ... [/Define_Tag] tags to determine what this tag actually does.

```
<?LassoScript
    define_tag('myCustomTag',
        -namespace='Ex_',
        -priority='none',
        -required='Name',
        -required='Value',
        -description='This is an example custom tag');

    /define_tag;
?>
```

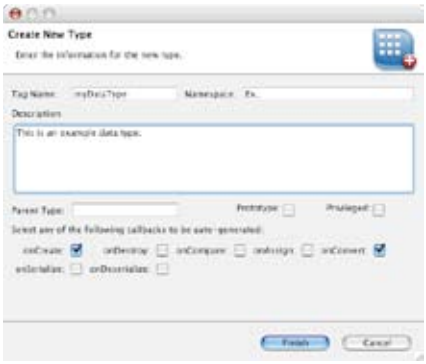
Note: See the ***Custom Tags*** chapter in the Lasso Language Guide for information about how to create custom tags.

To insert a custom data type template:

Select Type... from the contextual menu in the outline view. The dialog box shown in the figure below allows the attributes of the custom data type to be specified. These include:

- **Name** – The name of the custom tag. In this example myDataType is specified.
- **Namespace** – The namespace of the custom tag. In this example specifying Ex_ means the tag will be callable as [Ex_myDataType].
- **Description** – An optional description for the tag.
- **Options** – Many options for the tag can be specified including the parent type, -Privileged, and -Prototype.
- **Callbacks** – Templates for any of the available callback tags can be inserted. These include onCreate, onDestroy, onConvert, onAssign, onCompare, onSerialize, and onDeserialize.

Figure 5: Custom Data Type Template



After selecting Finish the type is inserted into the current file. The dialog above would create the following code (reformatted for easier reading). The contents of the tag can then be inserted between the [Define_Type] ... [/Define_Type] tags to determine what this data type actually does.

```
<?LassoScript
    define_type('myDataType',
        -namespace='Ex_',
        -description='This is an example data type.');

      define_tag('onCreate');

      /define_tag;

      define_tag('onConvert', -required='to');

      /define_tag;

    /define_type;
?>
```

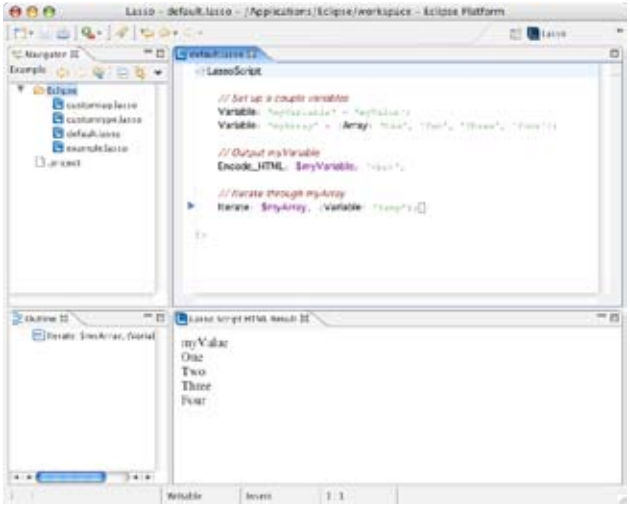
Note: See the ***Custom Data Types*** chapter in the Lasso Language Guide for information about how to create custom data types.

Code Folding

Each container tag in a LassoScript is accompanied by a disclosure triangle in the left margin of the editor window. This triangle can be used to temporarily hide the contents of any container tag. The container tag can be folded closed so the surrounding code can be read easier or re-opened to examine the code within.

The following figure shows the result of folding the [Iterate] ... [/Iterate] tag by clicking on its disclosure triangle.

Figure 6: Code Folding



Tag Completion

Since Lasso has over one thousand tags it can sometimes be difficult to recall exactly which tag needs to be inserted into a LassoScript. Lasso Studio provides automatic tag completion which recommends a list of possible tags based on the characters that have already been inserted into the document.

Tag completion is activated by typing ctrl-space (hold down the ctrl key and type a space) while typing within the editor. A menu of possible completions will appear. The desired tag can be inserted by selecting it with the mouse (double-click) or with the arrow keys and typing return.

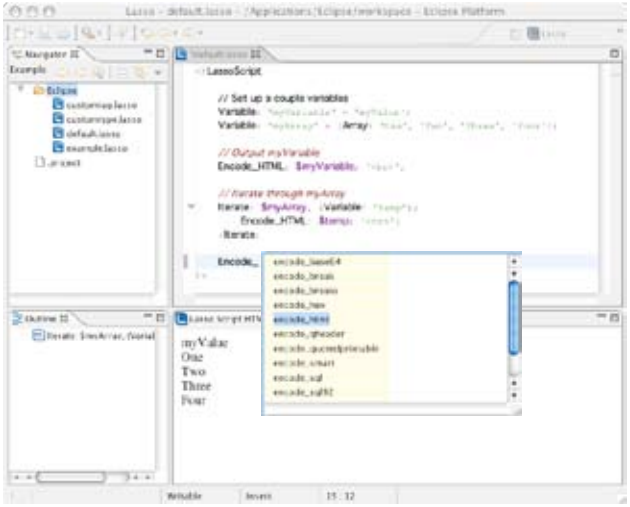
If a tag name has already been started when ctrl-space is typed then tags that begin with that string will be suggested first, followed by tags that include the string. Typing additional characters while the menu is showing will interactively reduce the number of options in the menu. Typing Field will suggest the tags [Field] and [Field_Names] first, but will also include tags such as [Required_Field] and [Search_FieldItem]. This allows a tag to be searched for even if you aren't sure how how the tag name begins.

The list of available tag completion options will also include any appropriate templates that have been defined. See the next section for details about using templates.

Tag completion will suggest the names of Lasso's built-in substitution and process tags and the names for custom tags which are defined in the current project. Tag completion will not suggest the names of member tags except for the members of the [Self] tag within a custom type definition.

In the following figure Encode_ has been typed into the editor and the completion menu activated by typing ctrl-space.

Figure 7: Tag Completion

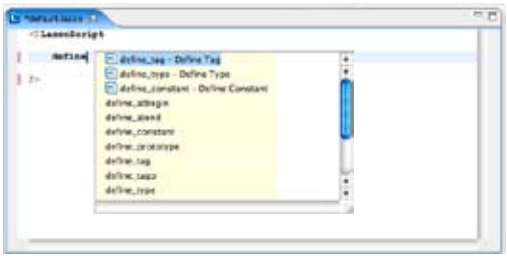


Templates

Templates provide more advanced tag completion including place-holders for user-defined values, database names, table names, and more. A template is activated the same way as the built-in tag completion options using ctrl-space within the editor. The available templates will be listed at the top of the menu that appears.

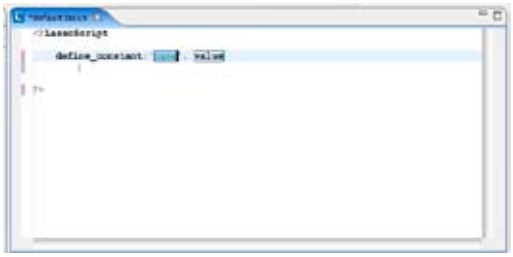
The following figure shows the result of typing Define and then ctrl-space within the editor. The resulting menu includes three templates for [Define_Constant], [Define_Tag] ... [/Define_Tag], and [Define_Type] ... [/Define_Type]. Below these template options the menu also includes traditional tag name completion options.

Figure 8: Template Selection



Selecting the [Define_Constant] template will insert it into the current document. The template includes two placeholders for name and value. The first placeholder will be automatically selected so a new value can be entered for it. Typing tab will move on to the next placeholder. This allows values for all the placeholders to be inserted without using the mouse.

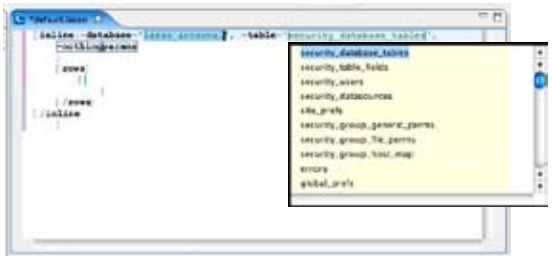
Figure 9: Template Placeholders



If a template includes a placeholder for a database name, table name, database action, variable name, or other values then a menu will be presented with appropriate values for that placeholder. The following figure shows the template for the [Inline] ... [/Inline] tag with the placeholder for the -Table parameter selected.

Note: The list of tables includes all of the tables that are accessible to Lasso. The table list is not automatically filtered based on the database that has been selected.

Figure 10: Template Menus

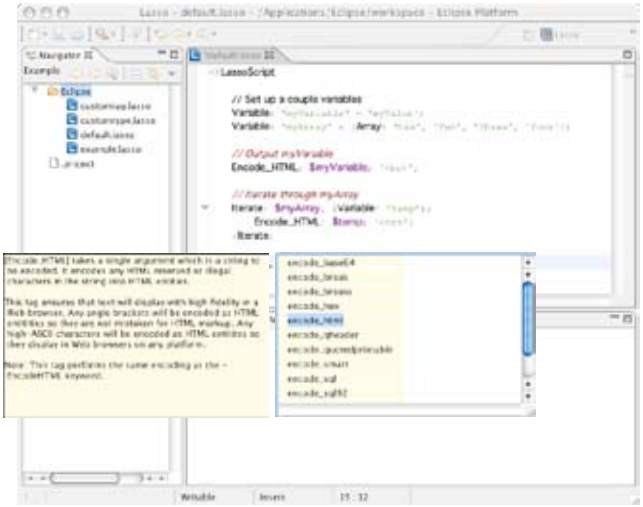


See the section on *Preferences* in this chapter for details about how to customize the built-in templates and how to create custom templates.

Lasso Reference

When the tag completion menu is visible Lasso Studio provides additional help by displaying the description of each tag in the list from the Lasso Reference. The following figure shows the Lasso Reference entry for [Encode_HTML].

Figure 11: Lasso Reference

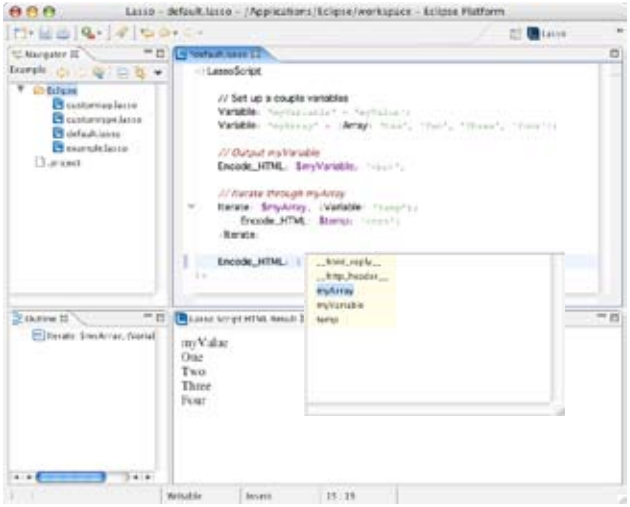


Variable Completion

In addition to tag names, Lasso Studio can also recommend page or local variable names within the editor. Variable completion is activated by typing $ or #. A list of possible variable names will be shown in a menu near the insertion point. The desired variable name can be selected with the mouse or with the arrow keys and then return.

The figure below shows the result of typing $ in the example file. The variables $myArray and $myVariable are shown in addition to several built-in page variables.

Figure 12: Variable Completion



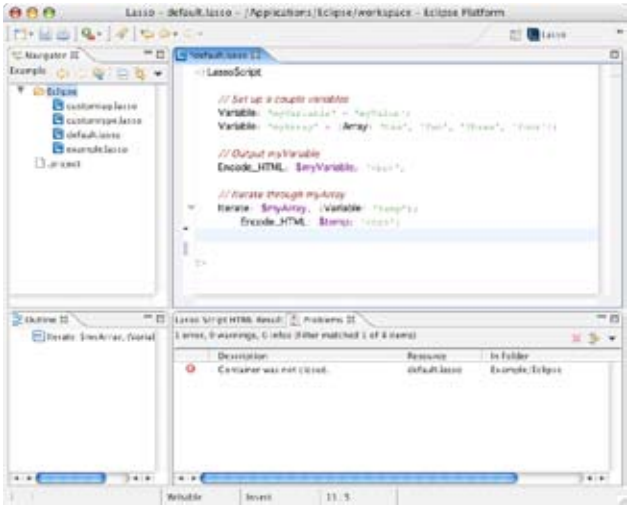
Syntax Checking

Lasso Studio will automatically check the syntax of any Lasso file that is open in an Eclipse editor. The syntax is checked when the file is opened and when the file is saved. Any issues that are found are reported in the built-in Problems pane of Eclipse.

If the Problems pane is not showing in the current workspace it can be opened by selecting the Window > Show View...> Others... and then selecting Basic > Problems. The Problems pane is normally paired with the LassoScript HTML Result pane at the bottom of the window.

The following figure shows the result of deleting the closing [/Iterate] tag and then saving the file. The syntax error is reported in the Problems pane as Container not closed.

Figure 13: Syntax Error



Correcting the issue by adding the closing [/Iterate] tag and re-saving the file will make the problem message disappear.

Note: Double-clicking on problems will open the file in which the error occurred if it is not already open.

What Errors Are Reported

Lasso Studio can report syntax errors in the Problems pane, but cannot report run-time errors until the LassoScript is actually executed (see the next chapter for details about how to debug these errors dynamically).

The following types of syntax errors are reported by Lasso Studio automatically:

- Missing closing container tags.
- Unterminated string literals including if the start and end quotes of a string don't match.
- Unterminated multi-line comments /* ....
- Missing semi-colons between tag calls.

The following types of errors cannot be reported by Lasso Studio without using the debugging features described in the next section:

- Mis-typed tag names or variable names. These can only be detected when the page is run since they could be defined globally or within an include.
- Data type errors such as using a member tag that does not exist for the current base type. These errors can only be caught when Lasso is running since a variable could be of any type depending on the LassoScript logic.
- Recursion or infinite loop errors.

Keyboard Shortcuts

Lasso Studio includes a number of keyboard shortcuts which can help accelerate the speed of working with your Lasso code. This section documents the keyboard shortcuts that Lasso Studio adds to Eclipse, some Eclipse shortcuts that are useful for Lasso Studio, and how to customize both Lasso Studio and built-in Eclipse shortcuts.

Lasso Studio Keyboard Shortcuts

The following table shows the keyboard shortcuts that Lasso Studio adds to Eclipse. These keyboard shortcuts can only be used when editing LassoScript files in the Lasso or Debug perspectives.

Table 1: Lasso Studio Keyboard Shortcuts

Command	Keyboard Shortcut
Add Block Comment	ctrl-shift-/ or command-shift-/
Content Assist / Auto Complete	ctrl-space or command-space
Format	ctrl-shift-F or command-shift-F
Open Definition	F3
Remove Block Comment	ctrl-shift-\ or command-\
Show Tooltip Description	F2
Toggle Line Command	ctrl-/ or command-/

Note: When two shortcuts are listed the first is generally for Windows and the second for Macintosh.

Eclipse Keyboard Shortcuts

Eclipse has many keyboard shortcuts. Some of the shortcuts which are specifically useful when working with Lasso Studio are shown in the following table.

Table 2: Eclipse Keyboard Shortcuts

Command	Keyboard Shortcut
Word Complete	alt-/ or ctrl-period
Lasso Script HTML Result Refresh	F5
Debug Step Into	F5
Debug Step Over	F6
Debug Step Return	F7
Debug Resume	F8
Key Assist	ctrl-shift-L or command-shift-L

Note: When two shortcuts are listed the first is generally for Windows and the second for Macintosh.

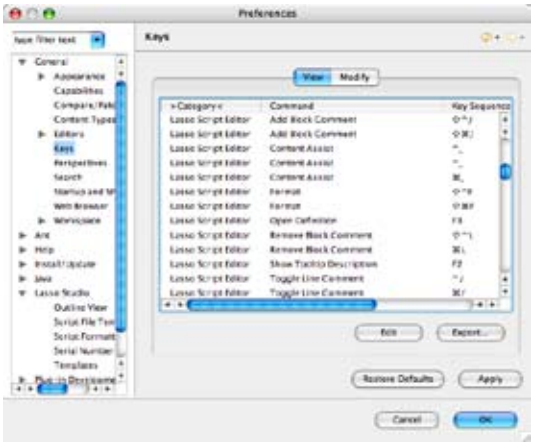
Customizing Keyboard Shortcuts

Many of the keyboard shortcuts within Eclipse can be modified. These include the shortcuts that Lasso Studio adds as well as many built-in shortcuts.

Access the preferences dialog box by choosing Preferences... from the Windows menu (or the application menu on Macintosh). Keyboard shortcuts can be modified in the General > Keys panel of the dialog box.

The following figure shows the keyboard shortcut panel scrolled to display the Lasso Studio specific shortcuts.

Figure 14: Keyboard Shortcuts Preferences



Preferences

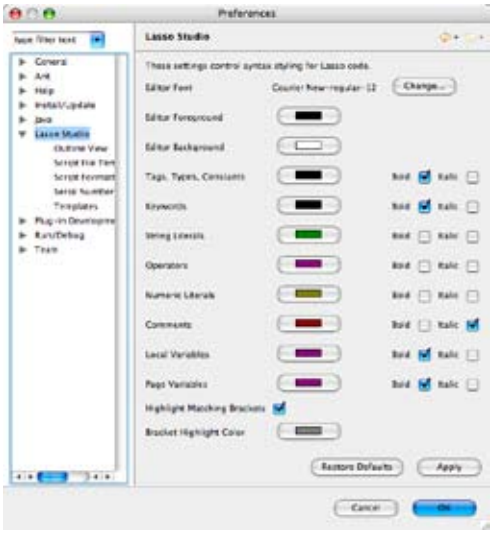
Lasso Studio adds a number of panels to the Eclipse preferences dialog box. Access the preferences dialog box by choosing Preferences... from the Windows menu (or the application menu on Macintosh). Each of the panels which Lasso Studio adds are described in detail below.

Lasso Studio (Syntax Coloring)

Selecting the Lasso Studio parent from the outline of available panels will show the syntax coloring preferences for Lasso Studio. These preferences allow different colors to be selected for all of the different elements that Lasso Studio recognizes. The Restore Defaults button can be used to restore the original settings for all the available options.

The Editor options control the font and default foreground and background colors for the editor. Colors and/or text styles can be set for Tags, Types, Constants, Keywords, String Literals, Operators, Numeric Literals, Comments, and Local and Page Variables. Finally, matching brackets can be highlighted when they are typed and the highlight color can be set.

Figure 15: Syntax Coloring Preferences



Outline View

Selecting the Lasso Studio > Outline View panel from the outline of available panels allows the elements that are shown in the outline view to be customized. The Restore Defaults button can be used to restore the original settings for all the available options.

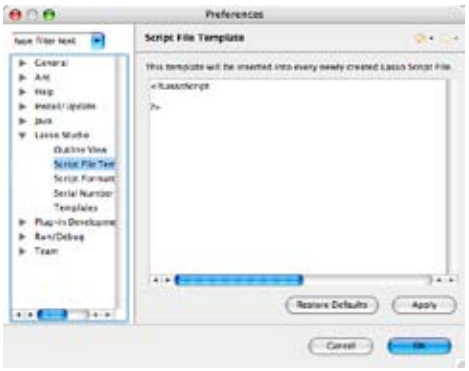
The outline view can contain any of a number of container tags, comments, and/or string literals.

Figure 16: Outline View Preferences

Script File Template

Selecting the Lasso Studio > Script File Templates panel from the outline of available panels allows the code that is inserted into new LassoScript files to be customized. The Restore Defaults button can be used to restore the original template.

Figure 17: Script File Template Preferences



Script Formatting

Selecting the Lasso Studio > Script Formatting panel from the outline of available panels allows the preferences for how Lasso Studio generates code and how Lasso Studio formats and re-formats code to be specified. The Restore Defaults button can be used to restore the original settings for all the available options.

- **Preferred Coding Style** sets the coding style to either prefer <?LassoScript ... ?> or square bracket [...] delimiters. This preference should be set to the coding style that you usually use on projects.
- **Preferred Parameter Style** sets the desired syntax to either colon syntax or parentheses syntax.

Note: These first two preferences should be set by every user to ensure that Lasso Studio uses their preferred coding and parameter style. The remaining preferences should only be modified if the default settings produce undesired results.

- **Ignore Blank String Literals** controls how Lasso Studio treats whitespace when it formats code. If this option is set then strings of whitespace between square brackets will be ignored when converting code to the LassoScript coding style.
- **Indentation Strings** sets the desired indentation style to use either tabs or from one to five spaces.
- **Quote Characters** sets the desired delimiter for string literals to either single quotes or double quotes.
- **Wrap at Column** sets the longest desired line length in characters. The default is 80 characters which means that lines of code will be wrapped to be less than 80 characters long.
- **Maximum Parameters per Line** sets the maximum desired number of parameters that should be left on one line when formatting [Inline] ... [/Inline] tags and other tags that have many parameters.
- **Maximum Plain Text Length** sets the maximum length of text which should be converted into string literals. If a string of plain text (text without any Lasso tags) exceeds this length when code is converted to use the LassoScript coding style then the <?LassoScript ... ?> delimiters will begin and end rather than using quotes to surround the string within the LassoScript.

Figure 18: Script Formatting Preferences

Serial Number

Selecting the Lasso Studio > Serial Numbers panel from the outline of available panels allows the serial number for Lasso Studio to be entered. This panel can be used to enter a purchased serial number after first trying out an evaluation version of Lasso Studio.

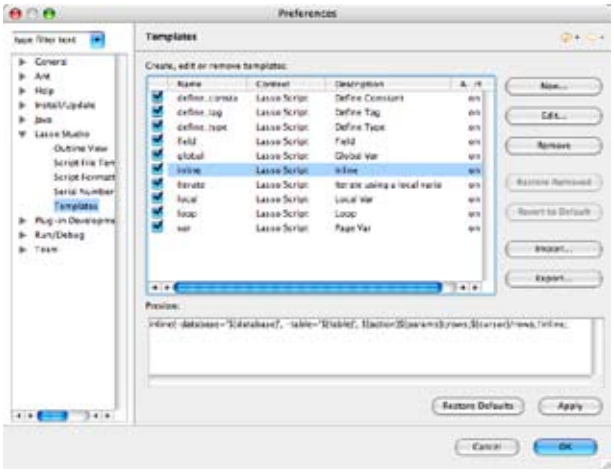
Templates

Selecting the Lasso Studio > Templates panel from the outline of available panels allows the templates which are inserted by the auto complete mechanism to be customized. The Restore Defaults button can be used to restore the original settings for all the available options.

The listing at the top of the panel shows the available templates and allows them to be easily activated and deactivated. Each template has a name, context (always Lasso Script), description, and auto enter value. When a template is selected the pattern for the template can be seen in the text area below.

Double-clicking on a template opens up the Edit Template dialog box which allows the template to be modified. The panel also allows new templates to be created, templates to be removed, and for templates to be imported and exported.

Figure 19: Templates Preferences



The pattern for the inline template is shown below. Each of the elements which are enclosed in ${...} is a placeholder that can be filled in by the user when they insert the template. Any name can be given to a placeholder.

```
inline(-database='${database}', -table='${table}', ${action}${params});rows;${cursor}/rows;/inline;
```

However, some placeholder names have special meanings and are called variables. The available variables can be accessed using the Insert Variable... button in the Edit Template dialog box. Some variables are special placeholders that show a menu of options while others simply insert the specified value when the template is used. Each of the variables is described in the table that follows.

Table 3: Template Variables

Variable Name	Description
action	Placeholder provides a menu of database actions.
cursor	Defines the cursor position after each of the placeholders has been fulfilled.
database	Placeholder provides a menu of available databases.
date	Inserts the current date
dollar	Inserts a dollar sign $
line_selection	Inserts the current selected lines.
table	Placeholder provides a menu of available tables. Note that tables are not filtered by database selection.
time	Inserts the current time.
user	Inserts the current user name.
word_selection	Inserts the current selected word.
year	Inserts the current year.

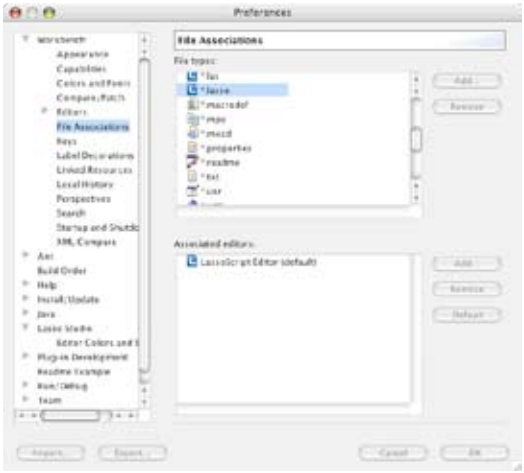
File Associations

This section describes how to use the Eclipse preferences to associate additional file types or file extensions with the LassoScript editor that is included with Lasso Studio. This allows .html files to be viewed with LassoScript syntax coloring.

To edit .html files in the LassoScript editor:

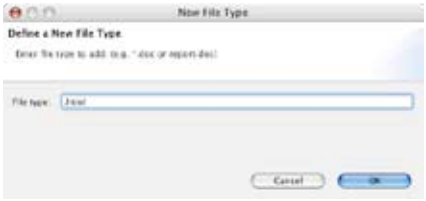
1 Open the Eclipse preferences dialog and select the Workbench > File Associations option from the list on the left. The following figure shows the preferences window with the file association for .lasso files selected.

Figure 20: Eclipse .lasso Preferences



2 To add a new file extension for .html. First, check to see whether .html is included in the File Types list. Select the Add... button to add it to the list. The following figure shows the new file type dialog box with .html entered.

Figure 21: New File Type

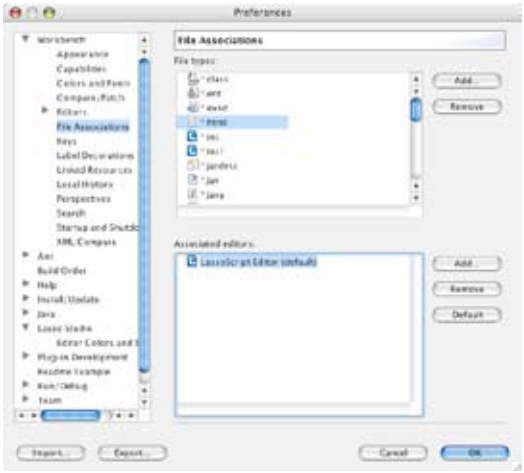


3 To add an editor association, make sure .html is still selected in the File Types list. Select the Add... button next to the Associated Editors... list. The following figure shows the Editor Selection dialog with the LassoScript Editor option selected.

Figure 22: Editor Selection

4 Now the .html extension is associated with the LassoScript editor. Double-clicking on a .html file in the navigator view will open the editor with LassoScript syntax coloring and the other features described in this chapter active.

Figure 23: Eclipse .html Preferences



6

Chapter 6
Running LassoScripts

This chapter describes how to run LassoScripts from within Eclipse.

- *Overview* provides an overview of how Lasso Studio runs scripts and how errors are reported.
- *Run Configuration* describes how to configure Lasso Studio to run LassoScripts.
- *Running Scripts* provides detailed information about running LassoScripts within Lasso Studio.
- *Additional Configuration* describes each section of the Create, Manage, and Run dialog box.

Overview

Lasso Studio can communicate with Lasso Professional in order to run a Lasso file from the Navigator view. The results of the LassoScript or any errors that occur during execution are reported in the LassoScript HTML Result view within Eclipse.

Running LassoScripts from within Eclipse makes the following activities possible:

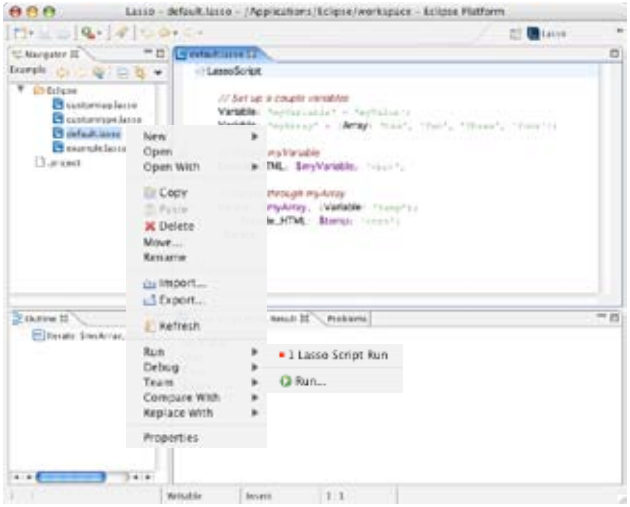
- Check Lasso files for runtime errors. Any runtime errors encountered will be reported automatically.
- Quickly run LassoScripts. Scripts can be used to run quick calculations, regular expressions, etc. without uploading pages to a Web server.
- Perform database actions on the server by running [Inline] ... [/Inline] tags.

Important: A file that is run from within Lasso Studio is actually executed within the associated Lasso Professional server. In particular, any database operations or file tags operations that are performed will occur just as if the page had been called through a Web browser.

The first time a file is run a Run Configuration must be established within Eclipse. This configuration tells Lasso Studio where to find Lasso Professional and provides a username and password for access. Lasso Studio requires a user within the Lasso Studio for Eclipse group or the site administrator username and password

Once a Run Configuration is established, the contextual menu in the Navigator can be used to run any Lasso file in the current project. The results of the run, either the HTML output or a runtime error report, are reported in the LassoScript HTML Result view.

Figure 1: Navigator Run Menu



Run Configuration

A Run Configuration is required for Lasso Studio to be able to communicate with Lasso Professional. The available configurations are listed in the Run sub-menu shown in the figure above. Initially, this menu will only contain the Run > Run... option which allows a new configuration to be established.

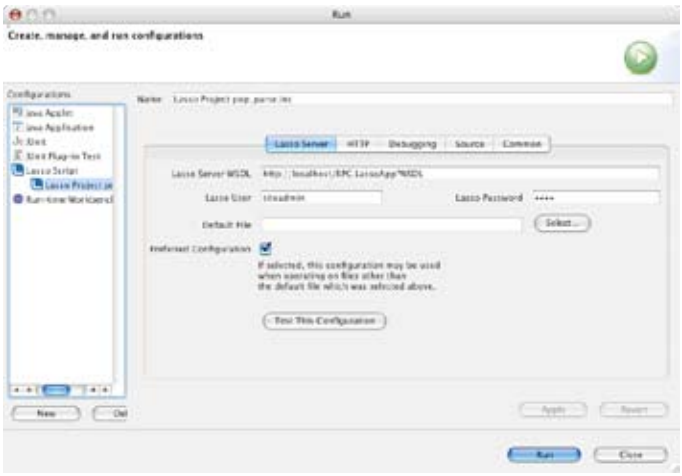
To establish a new run configuration:

1. Select the Run > Run... option from the contextual menu for a Lasso file within the navigator view.
2. On the Lasso Server tab of the dialog box, type the location of the Lasso server in the Lasso Server WSDL field. This URL should be specified as follows by default. If Lasso is hosted on a remote server the URL can be adjusted to reflect the DNS host name or IP address of the machine.

   http://localhost/RPC.LassoApp?WSDL

3. Enter a username and password into the Lasso User and Lasso Password fields. These should be either for a user within the Lasso Studio for Eclipse group or for the site administrator.
4. The Default File should reference any file within the current projects. Use Select... to choose a Lasso file. The Preferred Configuration checkbox should be checked. A preferred configuration can be used to run any Lasso files within the current projects.
5. Click the Test This Configuration button. If an error is reported double check the Lasso Server WSDL, Lasso User, and Lasso Password fields to make sure they contain valid values.
6. Select the Apply button and then the Close button to return to the main Eclipse window.

Figure 2: Create, Manage, and Run Configurations

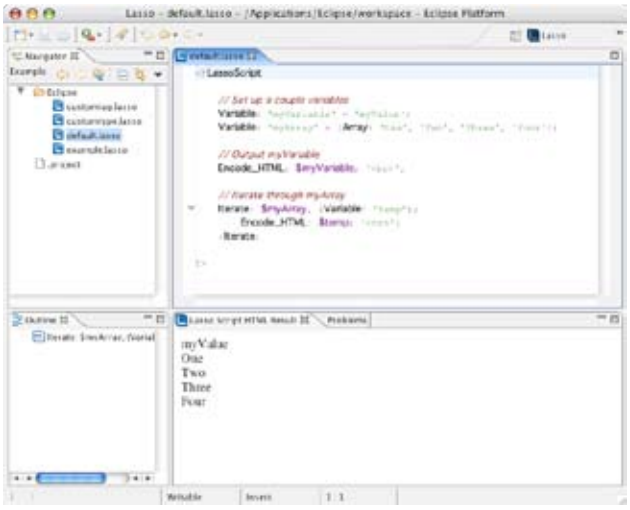


Note: The remaining tabs in this dialog can be left at their default values. The options presented in these tabs is described in detail in the *Additional Configuration* section later in this chapter.

Running Scripts

Once a run configuration has been created it can be used to run Lasso files from within the navigator view. Select the Run > LassoScript Run option from the contextual menu on a file. The results of running the file will be shown in the LassoScript HTML Result view. The following figure shows the results of running the example file in the editor.

Figure 3: LassoScript HTML Result



As long as the script completes normally the HTML that was generated by the Lasso file will be shown in the LassoScript HTML Response view. The result is exactly the same as if the script was called by URL and returned HTML to a Web browser.

If any files have not been saved before a script is run then a dialog box will be shown which lists all unsaved files and allows them to be saved before they are handed off to Lasso Professional.

Reporting Errors

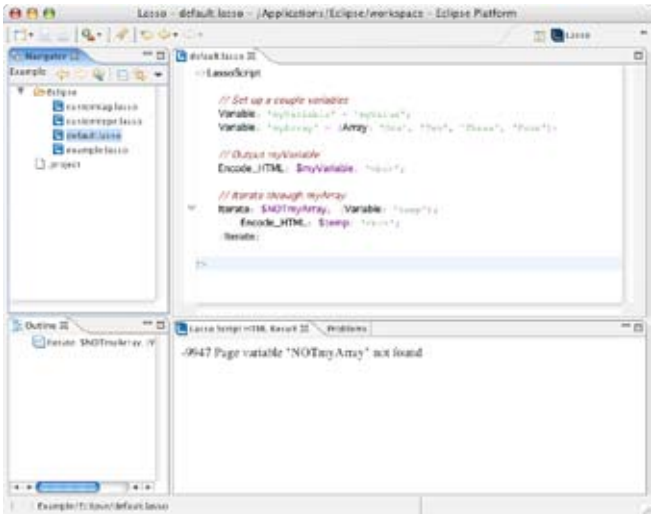
If a LassoScript generates a runtime error then it will be reported in the LassoScript HTML Response view in lieu of the HTML generated by the page. Lasso is unable to continue processing a Lasso file once it hits a runtime error.

The following figure shows the result of running the example file shown in the editor after the name of the variable reference $myArray has been changed to $NOTmyArray. The result is the following error message.

```
-9947 Page variable "NOTmyArray" not found
```

By correcting the error message and running the file again it is possible to create a page free of both syntax errors and runtime errors before it is uploaded to the Web server.

Figure 4: LassoScript Error



Note: Syntax errors that can be detected when the file is saved are reported in the Errors view. A file should always be free of syntax errors before it is run. However, if a file contains syntax errors or generates runtime errors they will be reported in the LassoScript HTML Response view after the file is run.

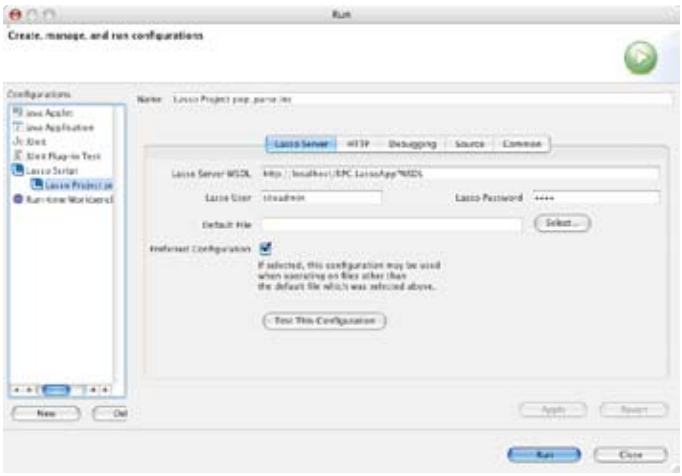
Additional Configuration

This section describes each of the tabs of the Create, Manage, and Run Configurations dialog box in Eclipse for the LassoScript > Lasso Project item. The settings in each tab other than the Lasso Server tab are optional and can usually be left set to their default values.

The dialog box includes the following tabs, each of which is described fully in the sections below.

- **Lasso Server** – Required configuration of where Lasso Professional is located and what username and password to use for authentication.
- **HTTP** – Allows additional headers to be sent with HTTP requests.
- **Debugging** – Configures the debugger.
- **Source** – Controls where Lasso Studio will search for source files.
- **Common** – Launch configuration preferences.

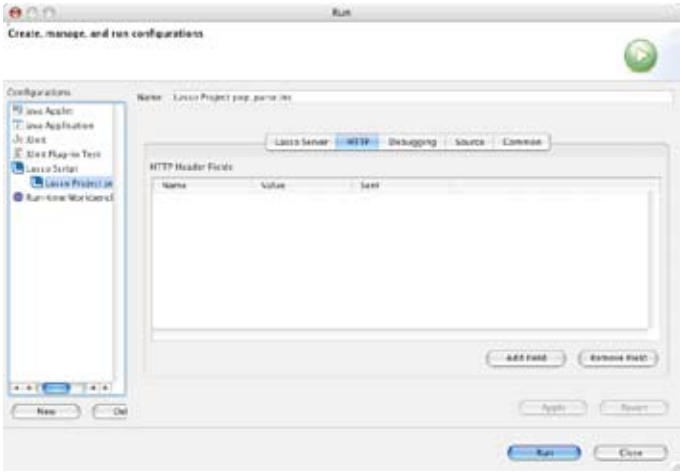
Figure 5: Run Configuration – Lasso Server



The Lasso Server tab controls the basic configuration of Lasso Studio including the location of the Lasso Server WSDL service and the Lasso User and Lasso Password which should be used to access Lasso Professional. The user should be a member of the Lasso Studio for Eclipse group or the site administrator.

The Default File should be set to a Lasso file in the current project. The Preferred Configuration checkbox should be checked. Any time any of the settings on this tab are changed the Test This Configuration button should be clicked to make sure there are no problems reaching Lasso Professional.

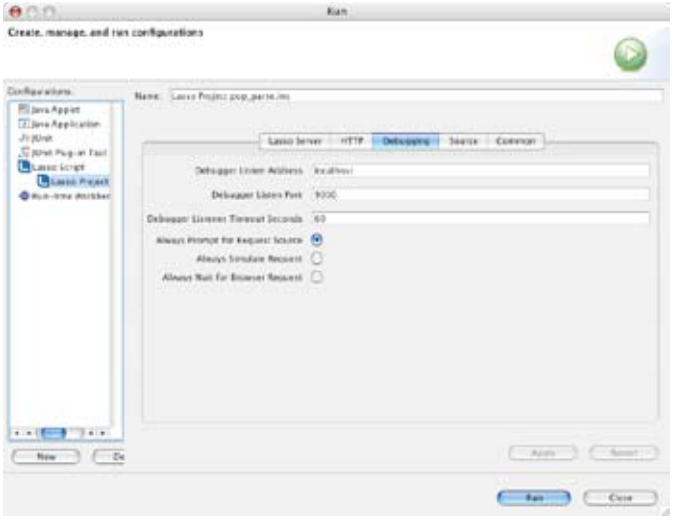
Figure 6: Run Configuration – HTTP



The HTTP tab allows extra HTTP headers to be specified. Each time a request is sent to Lasso these HTTP headers will be included. Select the New button to create a new header in the listing, then edit the values directly. The Sent column can be set to Always, Run, or Debug in order to only send certain headers when the Lasso file is being run or debugged.

For example, adding a header with name Debug and value True that will be sent on Debug requests can be checked within Lasso code using [If: (Client_Headers >> Debug)] ... [/If] in order to activate special code to run when a page is being debugged.

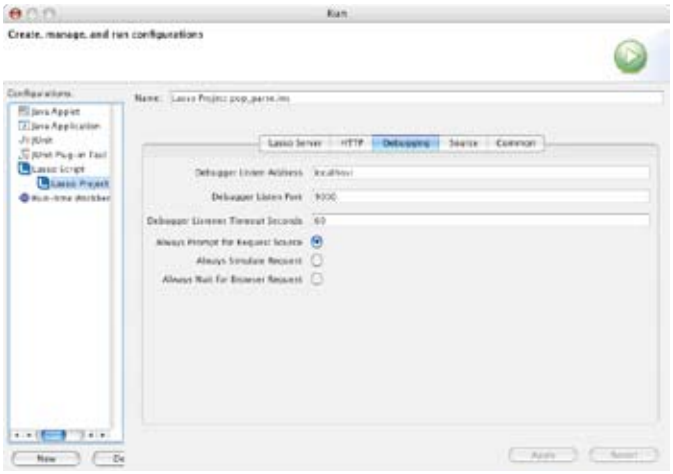
Figure 7: Run Configuration – Debugging



The Debugging tab allows several debugger preferences to be configured. The default values of these preferences should work for most installations. Full details about these preferences are provided in the *Debugging* chapter.

- **Debugger Listen Address** – The address on which Lasso Studio will listen for responses from the debugger. Set to localhost by default.
- **Debugger Listen Port** – The port on which Lasso Studio will listen. Set to 9000 by default.
- **Debugger Listen Timeout Seconds** - The timeout for responses from the debugger. Set to 60 seconds by default.
- **Always Prompt for Request Source** – If this option is checked then Lasso Studio will show a dialog box each time a file is debugged with each of the following two options.
- **Always Simulate Request** – If this option is checked then Lasso Studio will always simulate a Web browser request each time a file is debugged.
- **Always Wait for Browser Request** – If this option is checked then Lasso Studio will always wait for an actual Web browser request each time a file is debugged.

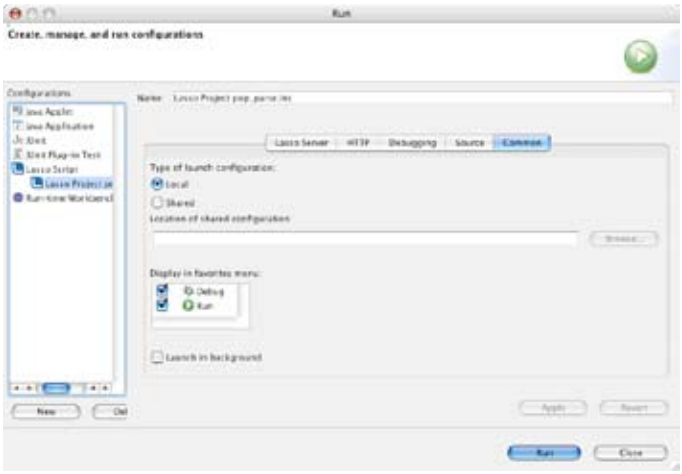
Figure 8: Run Configuration – Source



The Source tab specifies where Lasso Studio should look for Lasso files. If Lasso Studio ever reports that it is unable to find a source file then this tab should be checked to make sure that the path to the source file is included.

The entire current workspace can be added by selecting Add... and choosing Workspace in the resulting dialog box. Alternately, one or more projects can be added by selecting Add..., choosing Project in the resulting dialog box, and then selecting the projects to be added.

Figure 9: Run Configuration – Common



The Common tab includes several options that control how Lasso Studio runs or debugs files.

- **Type of Launch Configuration** – Should be set to the default value of Local. Shared configurations are beyond the scope of this documentation.
- **Display in Favorites Menu** – Controls whether the Debug or Run options appear in the Favorites menu. By default both these options should be checked.
- **Run in Background** – Determines whether feedback should be given interactively or not. By default this option should be unchecked so feedback will be provided.

7

Chapter 7
Debugging LassoScripts

This chapter describes using the debugging features of Lasso Studio including running LassoScripts from within Eclipse and using the interactive debugger to step through Lasso code.

- *Overview* provides a conceptual overview of using database snapshots in Lasso Studio.
- *Debug Configuration* provides a pointer to the *Run Configuration* section in the previous *Running LassoScripts* chapter where the steps required to configure Lasso Studio to either run or debug LassoScripts are described.
- *Debugging Interface* describes the views and editor features of the Debug perspective.
- *Triggers* describes how Lasso Studio either simulates Web server requests or installs a trigger for the page to be debugged and waits for an actual Web browser request.
- *Debugging Example* walks through an interactive debugging session including triggering the debugger, setting a breakpoint, resuming execution to the breakpoint, stepping through code, and modifying a variable while the page is running.

Overview

Lasso Studio integrates with the debugger in the Eclipse IDE allowing for interactive debugging of LassoScripts. The Lasso Studio debugger has the following features.

- Step-by-step debugging of Lasso files.
- Set breakpoints in Lasso files.
- View or modify the contents of variables while debugging.
- View the current HTML output of the page while debugging.
- Trigger debugging by simulating a Web request or by waiting for a request from an actual Web browser.

Debugger Workflow

The workflow of a debugging session is described below.

1. The debugger is activated for a file in Eclipse. One or more breakpoints can be set in the file.
2. Eclipse will prompt for the trigger method. The trigger can either be through a simulated Web request or can wait for an actual Web request from a Web browser.
3. The trigger request starts the interactive debugging session. The Eclipse perspective is set to Debug, the current file is opened in the editor, and the page is halted before any Lasso tags are executed.
4. The debugger controls are used to step through the page, run until the next breakpoint, step in or out of include files, etc. As each Lasso tag is executed the values for all watched variables is updated.

5 Once the Lasso page reaches the end the interactive debugger session is closed. The results of the Lasso page can be viewed in the Lasso Studio HTML Result window.

Only one interactive debugging session can be open at a time.

Debug Configuration

The debugger shares the same run configuration described in the *Running LassoScripts* chapter. Before using the debugger the steps in the *Run Configuration* section of that chapter should be followed.

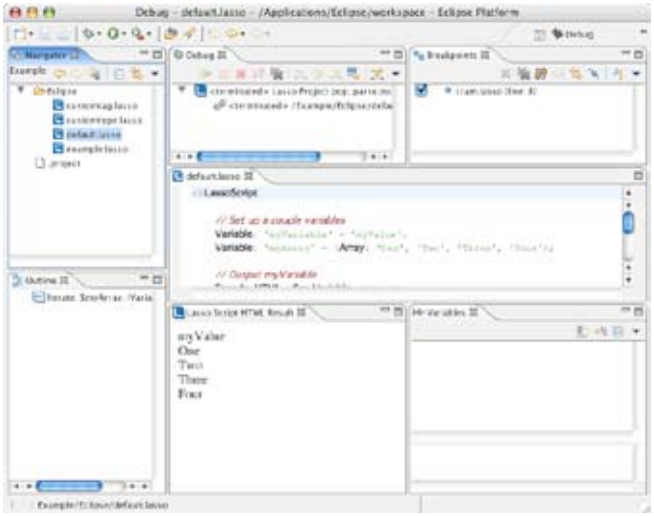
Debugging Interface

Debugging in Eclipse happens in the Debug perspective. Eclipse will automatically switch to this perspective when a debugging session is started.

The Debug perspective can also be selected manually by choosing the Debug option (with the bug logo) from the list of perspectives in the upper-right portion of the window. If Debug does not appear in the option then select Open Pespective > Other... from the Window menu and choose Debug from the list.

The following figure shows the basic interface elements of the Debug perspective. within Eclipse. This perspective defines a standard set of panes and views that allow LassoScripts to be edited and for syntax errors to be reported.

Figure 1: Interface Elements



The interface is split into seven views and editors which are described below.

- **Navigator** – The navigator shows the files that are open in the current project. A file can be opened in the editor by double-clicking on its name. The contextual menu on each file allows it to be run or debugged.
- **Debug** – The debug view shows information about the current file being debugged and contains the debugging navigation controls.
- **Breakpoints** - Contains a list of all the breakpoints that have been defined. Allows breakpoints to be turned on or off. Double-clicking on a breakpoint opens the file that contains it in the editor.
- **Editor** – The editor shows one or more open files. A different file can be selected by choosing a different tab.
- **Outline** – The outline shows a hierarchical view of the container tags within the current file shown in the editor. A given container tag can be selected by double-clicking on its name in the outline.

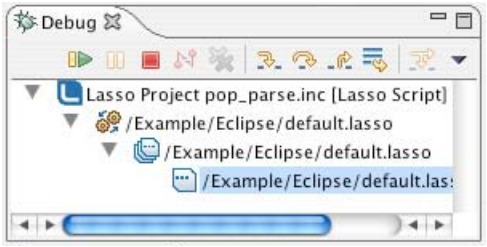
- **LassoScript HTML Result** – The LassoScript HTML Result view displays the result of running the page.
- **Variables** – Shows the variables defined on the current page. Allows their values to be viewed or modified while a Lasso page is being debugged.

These interface elements can be moved relative to one another and saved as a new perspective. This default arrangement can be restored by reselecting the Debug perspective.

Debug View

The debug view includes a series of controls across the top which are used to control how a LassoScript is executed while it is being debugged. Each control is described in detail below.

Figure 2: Debug View



The controls are as follows. Each of these controls also appears in the Run menu and has a keyboard equivalent which can speed up repeated use.

Resume – Allows execution of the current Lasso page to proceed to the next break point. If no breakpoints are set then the page will execute until it has completed.

Suspend – Allows a page that is running to be suspended. This can be useful in pages that take many seconds to execute.

Terminate – Terminates the current debugging session. The page is not allowed to finish executing. Similar to calling [Abort] within a Lasso page.

Remove All – Clears the Debug view. This control is only active when there is no current debugging session.

Step Into – Allows the execution of a single Lasso tag call. If the tag call is an [Include] or [Library] then the debugger will step into the included file so that the Lasso code contained within can be debugged.

Step Over – Allows the execution of a single Lasso tag call. Steps over [Include] or [Library] tags running all of the included Lasso code as a single step.

Step Return – Allows the execution of the current Lasso page until its end. If the current Lasso page is called by an [Include] or [Library] tag then execution will be halted within the including Lasso page immediately after the tag.

Note: The Disconnect, Drop to Frame, and Step Filters controls are beyond the scope of this documentation. Consult the Eclipse documentation for details about how these options work.

Breakpoints View

The breakpoints view shows all of the breakpoints that have been defined. The checkbox allows individual breakpoints to be turned on and off. The Remove All control allows all breakpoints to be cleared. Double clicking on a breakpoint will open the associated file in the editor with the breakpoint showing. See the following section for details on how the editor shows breakpoints.

Figure 3: Breakpoints View

Note: Other breakpoint controls are beyond the scope of this documentation. Consult the Eclipse documentation for details about how these options work.

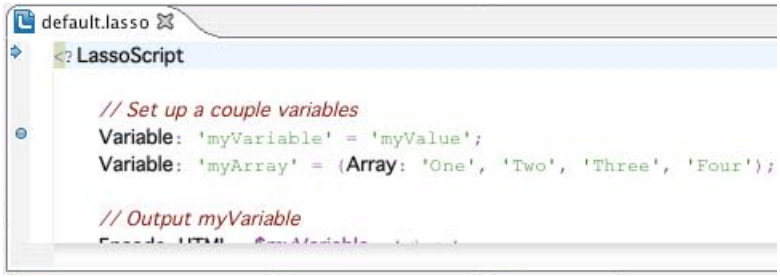
Editor

The editor view provides the same view of Lasso code as it does in the Lasso perspective described in previous chapters. See the *Using Lasso Studio* chapter for full details about its features.

Breakpoints are shown in the editor view in the far left column. A small blue circle shows a line that has a breakpoint set on it. Clicking on a circle (or where the circle would be in a blank line) will toggle the breakpoint on or off for that line. The following figure shows a breakpoint set on the first [Variable] line.

The current Lasso tag is indicated by a small blue arrow in the far left column. The arrow indicates the line that will next be executed if a step or resume control is used. The following figure shows the blue arrow pointing to the opening <?LassoScript delimiter indicating that no tags in the file have yet been executed.

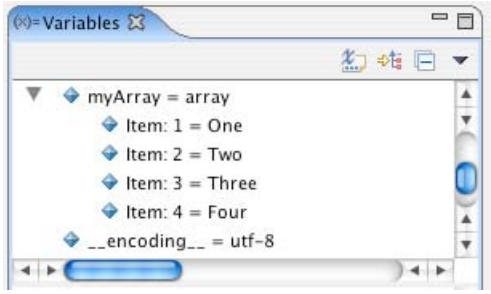
Figure 4: Editor



Variables View

The variables view shows all of the variables that are defined for the current executing Lasso page. As new variables are created they will be added to this view and as variables' values change it will be reflected in this view. The view includes several internal variables like __encoding__ and __http_header__ in addition to user-defined variables like myArray.

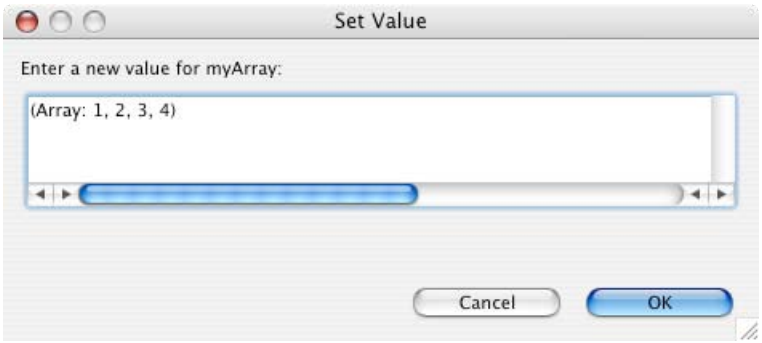
Figure 5: Variables



The figure above shows the $myArray variable expanded to show its elements. The value of a variable can be modified by double-clicking on the variable name. For example, clicking on the myArray = array line will allow the value of $myArray to be modified. Clicking on one of the Item 1 = One lines will allow one value within the array to be modified.

The Set Value dialog box allows a new value for a variable or an element of a compound data type to be specified. The dialog below show how to set the variable $myArray to a new value of (Array: 1, 2, 3, 4). Notice that the value can be any valid Lasso expression.

Figure 6: Set Value



In order to set a variable to a string value it is necessary to surround it by quotes within the Set Value dialog box.

```
"New String Value"
```

Triggers

There are two ways to trigger a page for an interactive debugging session. By default Lasso Studio will ask which method to use each time the Debug > LassoScript Debug option is selected from the context menu of a Lasso page in the navigator view. The Select Request Source dialog box is shown in the following figure. The options are described below.

Figure 7: Select Request Source

- **Simulate Request** – This option simulates a request for the Lasso page that will be debugged and triggers it immediately. Any extra headers that are specified in the current run configuration are sent along with the generated HTTP request. This should be the default option unless a Web browser request is needed.
- **Wait for Web Browser** – This option puts a trigger on the Lasso page to be debugged and then waits until a Web browser request comes in for that page. This allows a Lasso page to be debugged within the actual environment it will be called including the proper cookies from the Web client, browser type, authentication, etc.

  The URL for a trigger is dependent upon what folders the file is in within the Lasso Project. For example, a file default.lasso within an Eclipse folder in the current project could be loaded with this URL in a Web browser.

  ```
  http://localhost/Eclipse/default.lasso
  ```

Once the trigger URL is is loaded in the Web browser the Eclipse application will activate with the debugging session. Debugging can then proceed as with a simulated request. If the debugging session completes in time then a result will be returned to the browser (though most browsers will time out within a minute or so).

- **Cancel Debug Session** – This option cancels the debugger operation.

One of the options can be sent as a default in the run configuration on the Debugging tab of the Create, Manage, and Run Configuration dialog box which is discussed in the *Additional Configuration* section of the *Running LassoScripts* chapter.

Debugging Example

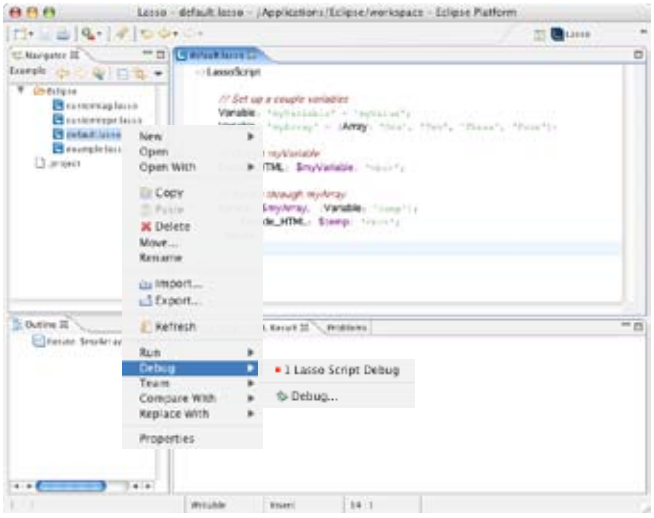
This section provides an example of debugging a page interactively. The default.lasso example file is debugged showing how to perform the following operations.

- The debugger is launched on an example file.
- A breakpoint is set in the file and the file is run until it hits the breakpoint.
- The step control is used to step over a Lasso tag and see the result.
- The value of a variable is examined and changed using the variable view.
- Execution of the page is allowed to complete using the new value for the variable.

This example should serve as a simple road map of how to interactively debug a file.

1 The example starts with the Lasso perspective visible in the Eclipse workspace. The file default.lasso is being edited. The Debug > LassoScript Debug option is chosen from the contextual menu on the default.lasso file icon in the navigator view.

Figure 8: Lasso Perspective

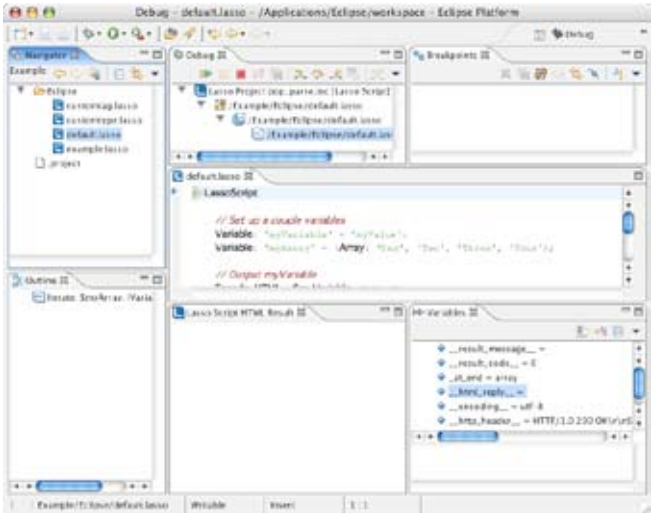


2 The Select Request Source dialog box is shown and the Simulate Request option is chosen. This will trigger the debugging session immediately.

Figure 9: Select Request Source

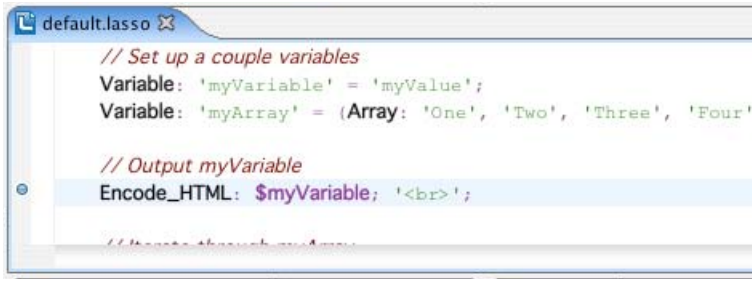
3 The Eclipse work space is changed to the Debug perspective. The file default.lasso is still visible in the editor pane. The blue arrow next to the <?LassoScript delimiter indicates that Lasso Studio is ready to debug the file, but has not yet executed any tags.

Figure 10: Debug Perspective



4 At this point a breakpoint is going to be set in the default.lasso file so that it can be run to that point without stepping. A breakpoint is set on the first line with [Encode_HTML] by double-clicking in the far left column. The blue circle indicates that the breakpoint has been set and a new line appears in the breakpoints view as well.

Figure 11: Set a Breakpoint



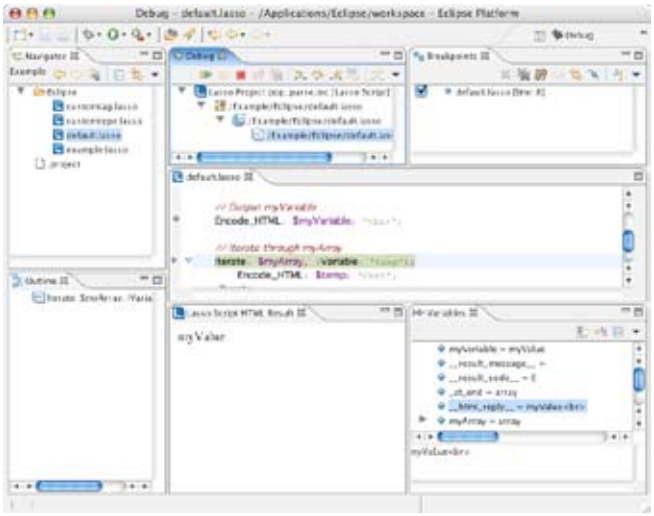
5 Now the resume control  is selected from the debug view. This will start the execution of Lasso tags within the file until the first breakpoint is set. In the case of this example, both variable definitions will be executed and then Lasso will pause just before executing the [Encode_HTML] tag.

Figure 12: Execution Halted at Breakpoint

6 Now the step over control  is used to advance Lasso past one tag. The [Encode_HTML] tag is executed and its value is placed into the HTML result for the page that is being assembled. The result can be seen in the LassoScript HTML Result view in the figure below.

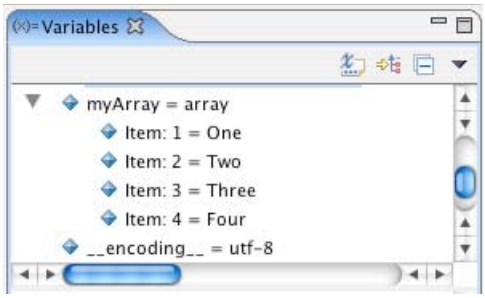
Also note that the variable $myVariable and its value is visible in the variable view and that the breakpoint that was set is visible in the breakpoints view.

Figure 13: Result of Stepping



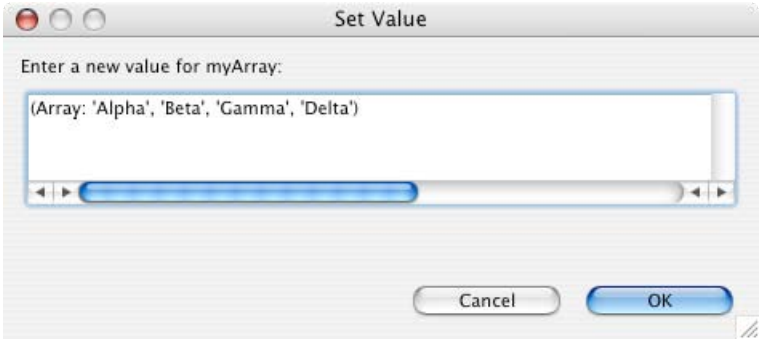
7 The debugger is now poised to start executing the [Iterate] ... [/Iterate] tags with the values from the $myArray variable. The variable view with the value for this variable is shown in the following figure.

Figure 14: Variables View



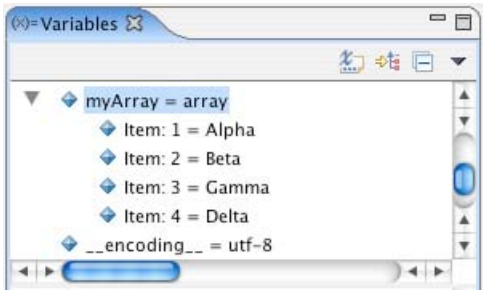
8 Double-clicking on the line with the name of the variable $myArray will bring up the dialog box for changing its value. This is shown in the following figure with a new expression defining an array that contains the names of Greek alphabetic characters rather than English numbers.

Figure 15: Set Value



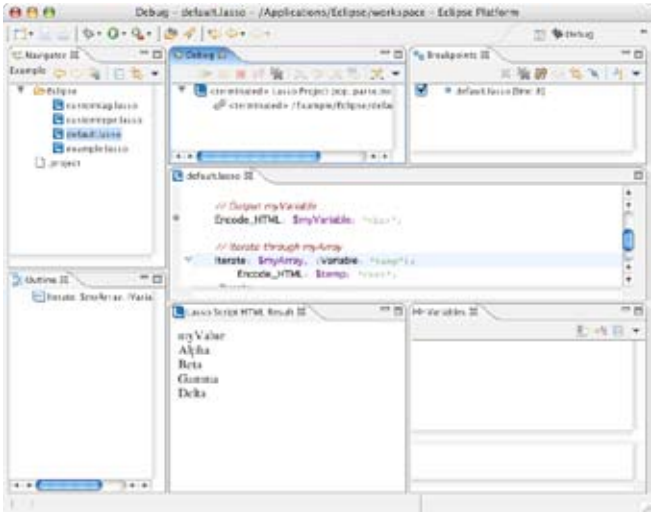
9 Now the variable $myArray is shown to have this new value in the variable view. As the page execution proceeds past this point this new value of the variable will be used in favor to the one it was given in the actual code.

Figure 16: Variables View



10 Now the resume control  is selected from the debug view again. Since there are no more breakpoints in the Lasso file the execution will proceed until the end. The final results of the execution are shown in the figure below. Notice that the results reflect the new value of the variable $myArray.

Figure 17: Final Results



A

Appendix A
License Agreement

PLEASE READ THIS LICENSE CAREFULLY BEFORE INSTALLING OR USING LASSO SOFTWARE. BY DOWNLOADING, INSTALLING, AND/OR USING THE SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS LICENSE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, YOU ARE NOT AUTHORIZED TO DOWNLOAD, INSTALL, AND/OR USE THIS SOFTWARE.

1 **License** – The Lasso products, applications, components, modules, application programming interfaces (APIs), and programming language including Lasso Studio for Eclipse, Lasso Service, Lasso data source connectors, Lasso Web server connectors, Lasso modules, Lasso Site Administration interface, and Lasso Dynamic Markup Language (Lasso), whether on disk, in read only memory, or on any other media or networked storage device (the "Lasso Software") and related documentation (the "data") are licensed to you by LassoSoft, LLC (the "Author"). You own the disk on which the Lasso Software and data are recorded but the Author and/or the Author's Licenser(s) retain title to the Lasso Software and related documentation.

   This License allows you to install and use a single copy of the Lasso Software and data on a single computer and make one copy of the Lasso Software and data in machine-readable form for backup purposes only. You must reproduce on such copy the Author's copyright notice and any other proprietary legends that were on the original copy of the Lasso Software and related documentation. You may transfer all your license rights in the Lasso Software and related documentation, the backup copy of the Lasso Software and related documentation, and copy of this License to another party, provided the other party reads and agrees to accept the terms and conditions of this License and a transfer of ownership letter, listing contact information for both parties, is sent to the Author.

2 **Upgrading** – In order to facilitate porting existing solutions to the current version of the Lasso Software, this License allows the prior version and the current version of the software to be run concurrently for a maximum of 45 days from the purchase date of the Lasso Software.

3 **Evaluation Versions** – Evaluation versions are provided for one-time 30-day evaluation and initial product testing use. Evaluation versions are not licensed for use for extended development.

4 **Restrictions** – The Lasso Software and related documentation contains copyrighted material, trade secrets and other proprietary material and in order to protect them you may not decompose, reverse engineer, disassemble or otherwise reduce the Lasso Software to a human-perceivable form without the Author's written permission.

5 **Termination** – This License is effective until terminated. You may terminate this License at any time by destroying the Lasso Software and related documentation and all copies thereof. This License will terminate immediately without notice from the Author if you fail to comply with any provision of this License. Subject to the terms of the Upgrading section above, this License will terminate upon upgrading, whether via free or paid upgrade, to a newer version of the Lasso software. Upon termination you must destroy the Lasso Software, and related documentation and all copies thereof.

6 **Disclaimer of Warranty on Lasso Software** – You expressly acknowledge and agree that use of the Lasso Software and related documentation is at your sole risk. The Lasso Software and related documentation are provided AS IS and without warranty of any kind and the Author EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE AUTHOR DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE LASSO SOFTWARE WILL MEET YOUR REQUIREMENTS, OR THAT THE OPERATION OF THE LASSO SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT DEFECTS IN THE LASSO SOFTWARE WILL BE CORRECTED.

   FURTHERMORE, THE AUTHOR DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE LASSO SOFTWARE AND DATA OR RELATED DOCUMENTATION IN TERMS OF THEIR CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY THE AUTHOR OR A REPRESENTATIVE AUTHORIZED BY THE AUTHOR SHALL CREATE A WARRANTY OR IN ANY WAY INCREASE THE SCOPE OF THIS WARRANTY. SHOULD THE LASSO SOFTWARE PROVE DEFECTIVE, YOU (AND NOT THE AUTHOR OR A REPRESENTATIVE AUTHORIZED BY THE AUTHOR) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU.

7 **Limitation of Liability –** UNDER NO CIRCUMSTANCES INCLUDING NEGLIGENCE, SHALL THE AUTHOR BE LIABLE FOR ANY INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES THAT RESULT FROM THE USE OR INABILITY TO USE THE LASSO SOFTWARE OR RELATED DOCUMENTATION AND DATA, EVEN IF THE AUTHOR OR AN AUTHORIZED REPRESENTATIVE OF THE AUTHOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU. In no event shall the Author's total liability to you for all damages, losses, and causes of action (whether in contract, tort (including negligence) or otherwise) exceed the amount paid by you for the Lasso Software and data.

8 **Complete Agreement** – This License constitutes the entire agreement between the parties with respect to the use of the Lasso Software, related documentation and data, and supersedes all prior or contemporaneous understandings or agreements, written or oral, regarding such subject matter. No amendment to or modification of this License will be binding unless in writing and signed by the Author or a duly authorized representative of the Author.

B

Appendix B
Glossary

This appendix contains definitions for many terms used in this book. If an unknown term does not appear in this glossary, see also the *Glossary* appendix in the Lasso Setup Guide.

Breakpoints

Contains a list of all the breakpoints that have been defined. Allows breakpoints to be turned on or off. Double-clicking on a breakpoint opens the file that contains it in the editor.

Container Tags

Container tags are delimited by square brackets and are comprised of both an opening and a closing tag. Much like in HTML markup, the area between the opening and closing tags is modified by the container tag. Container tags in Lasso perform database actions, hide code from the Web browser, perform conditional logic, and provide error control. Some container tags include:

```
[File_Control] [/File_Control]  [If] [Else] [/If]
[Inline] [/Inline]              [Loop] [/Loop]
[While] [/While]
```

Debug

The debug view shows information about the current file being debugged and contains the debugging navigation controls.

Editor

A text editor that displays a code file. Editors require that files be saved explicitly. Editors will display Lasso code with syntax coloring and disclosure triangles so portions of the code can be collapsed. Multiple editors can be open at once with tabs to select between them.

Form Action

In an HTML form, the form action defines how the form will be processed on the server. In order to work with Lasso, the form action needs to specify a valid Lasso response page that will be used as the response page for the form.

```
<form action=”/ResponsePage.Lasso” method=”POST”>
    <input type="hidden" name="-Response" value="Response.lasso">
    …
</form>
```

LassoScript HTML Result

The LassoScript HTML Result view displays the result of running the page. It includes one tab that shows the source code of the result and one tab that shows a rendered version of the HTML result.

Lasso 8.5

The product line produced by LassoSoft used to build and serve data-driven Web sites. This includes Lasso Professional Server, Lasso Development Studio, and Lasso Professional Developer.

Lasso Professional

The Lasso product used to serve data-driven Web sites. Lasso Studio includes a single-user license of Lasso Professional 8.5 for testing data-driven Web sites.

Lasso Studio

A Lasso product used to build and test data-driven Web sites to be served by Lasso Professional 8.5. Comes in Dreamweaver MX 2004 and Adobe GoLive CS editions.

Lasso

Lasso is an abbreviation for Lasso Dynamic Markup Language. Lasso is Lasso's tag-based markup language. It consists of container, process, member, and substitution tags which are delimited by square brackets and are processed when a Lasso-based page is served by your Web Server. Lasso also consists of command and action tags which are preceded by a hyphen. Command and action tags are contained in HTML form inputs or URLs and are processed when the form is submitted or URL is loaded from your Web server. LassoScript is an alternate form of Lasso syntax that is not delimited by square brackets.

Navigator

The navigator shows the files that are open in the current project. A file can be opened in the editor by double-clicking on its name.

Outline

The outline shows a hierarchical view of the container tags within the current file shown in the editor. A given container tag can be selected by double-clicking on its name in the outline. The contextual menu in the outline view can be used to insert custom tag or custom data type templates.

Perspective

The current arrangement of views and editors in the workbench is called the current perspective. Perspectives can be saved and recalled to switch between several different custom configurations.

Problems

The Problems view displays any syntax errors that are reported by Lasso Studio. Double-clicking on a problem line will open the containing file in the editor.

Process Tags

Process tags are delimited by square brackets and appear much like substitution tags except that they do not return any value to the page which will be shown to the user. Process tags include the following:

[Email_Send] [HTML_Comment]
[Log] [Session_Start]
[Variable_Set]

Project

Eclipse handles files and folders in projects. A project is a collection of files and folders that are managed by Eclipse. Eclipse has a great deal of flexibility for file management. Files can either be edited directly within the file system or can be imported into Eclipse and stored internally. Eclipse can also work with version control systems such as CVS.

Response Page

The Lasso page which Lasso displays to the user after a successful action. The response page can be specified within the form action of a preceding form. The Site Builder automatically creates the appropriate response pages for each database action.

Substitution Tags

Substitution tags are the most common tags in Lasso's tag-based markup language Lasso. They are delimited by square brackets and are replaced by a generated value as Lasso processes a Lasso page. For instance, the following tag will be replaced by the value for the field Company in the current database:

```
[Field: 'Company']
```

Substitution tags can also be used as attributes for Lasso tags. For instance, the following substitution tag adds two values, each defined by a field placeholder substitution tag:

```
[Math_Add: (Field: 'Total'), (Field: 'Shipping')]
```

Tag

Lasso uses a tag-based language called Lasso. Command and action tags can be inserted into HTML form inputs, URLs, or [Inline] tags:

```
<input type="hidden" name="-Database" value="[Database_Name]">
```

Substitution, container, and process tags are surrounded by square brackets and appear within the HTML source of a Web page:

```
[Field: 'Field Name']
```

Variables

Shows the variables defined on the current page. Allows their values to be viewed or modified while a Lasso page is being debugged.

View

A view is a part of the workbench that displays information. Some views display file listings or an outline view of a code files. Other views display syntax errors, runtime errors from the debugger, or the results of executing a LassoScript. Some views allow information to be changed and any changes are saved automatically.

Workbench

The main window of Eclipse is called the workbench. All of the views and editors that present information and allow files to be modified are displayed as parts of the workbench.

C

Appendix C
Index

T

U

V

W