1

Introducing SilkTest

In this chapter

This tutorial contains the following sections:

The Benefits of Automated Testing

Manually testing software is a time-consuming and often tedious process, one which cannot guarantee consistency of testing across releases and across platforms. Additionally, time constraints often do not afford us the luxury of being able to manually test and retest our applications before they are released. Inevitably the question remains, "Did any critical bugs go undetected?"

Automating your testing leads to a better use of your resources. Skilled testers are freed up to put more effort into designing better tests while machines that would otherwise lie idle overnight can be used to run unattended automated tests. The benefits of automating software testing with SilkTest are many:

- Providing more coverage of regression testing.
- Reducing the elapsed time for testing, getting your product to market faster.
- Improving productivity of human testing.
- Improving the re-usability of tests.
- Providing a detailed test log.

The layered approach

The most common approach to any testing is the layered approach. The layered approach includes three types of tests:

Operability Tests which examine each object, verifying specific properties of the object such as: state, size, caption and contents.

Functionality Tests which examine the behavior of a group of objects that together provide a specific feature to the end user. This includes looking at a dialog as a collection of objects and verifying the functionality provided. It can also include verifying the interaction between objects. For example, verifying that a text box is enabled when a check box is checked.

System Tests which examine how the application under test (AUT) interacts with other software or hardware products within the software environment.

Other types of tests

Other types of tests that may be performed using SilkTest include:

Regression Tests which run existing tests on new versions of a program.

Error Tests which verify the system’s response to error conditions.

Stress Tests which measure the system’s response to repetitive or large amounts of data.

White-Box vs. Black-Box Tests Where white-box testing places the focus on the internal structure of the software (the code) while black-box testing views the software from the end-user perspective and is unaware of the underlying code.

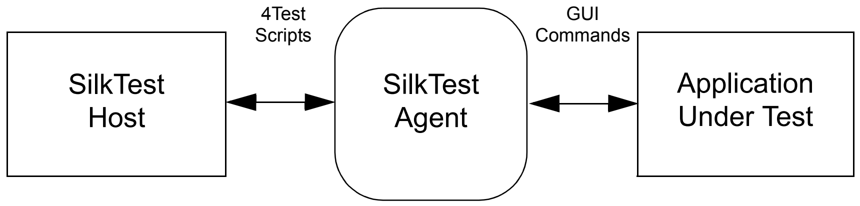
Understanding SilkTest Basics

How SilkTest works

Applications are composed of graphical user interface (GUI) objects such as windows, menus and buttons that a user manipulates using a keyboard and a mouse to initiate application operations. SilkTest interprets these objects and recognizes them based on the class, properties and methods that uniquely identify them. During testing, SilkTest interacts with the objects to submit operations to the application automatically, simulating the actions of a user, and then verifies the results of each operation. The simulated user, SilkTest, is said to be *driving* the application.

SilkTest consists of two distinct components that execute in separate processes:

- The SilkTest *Host* software
- The SilkTest *Agent* software



The SilkTest Host

The host software is the SilkTest component you use to develop, edit, compile, run and debug your test scripts and testplans. The machine that runs this component is often referred to as the *host machine.*

The SilkTest Agent

The SilkTest Agent is the component of SilkTest that interacts with the GUI of your application. The Agent translates the commands in your 4Test scripts into GUI specific commands, driving and monitoring the application you are testing. The Agent can run locally on the same machine on which the Host is running or, in a networked environment, any number of Agents can run on remote machines. In a networked environment, the machine that runs the Agent is often referred to as the *remote machine.*

How SilkTest records user actions

Before you begin creating and running test scripts, you create a repository of information about your application to be used by SilkTest. This repository includes descriptions of the GUI objects that comprise your application. Based on the properties and methods SilkTest associates with these objects, SilkTest can recognize the actions performed on them and intelligently record those actions into your test script using the 4Test language.

The table below provides a sample of what SilkTest records in the 4Test language when you complete a particular action.

What you do	What SilkTest records
You pick a menu item	`Pick`
You check a check box	`Check`
You uncheck a check box	`Uncheck`
You write text in a text field	`SetText`
You scroll a scrollbar to the maximum position	`ScrollToMax`
You select an item from a list box	`Select`
You close a dialog box	`Close`
You set the main window active	`SetActive`
You select a radio button from a group	`Select`

The 4Test language

4Test is an object-oriented fourth-generation language (4GL) designed specifically with the needs of the QA professional in mind. 4Test's powerful features are organized into three basic kinds of functionality:

- A robust library of object-oriented classes and methods that specify how a testcase can interact with an application's GUI objects.
- A set of statements, operators and data types that you use to introduce structure and logic to a recorded testcase.
- A library of built-in functions for performing common support tasks.

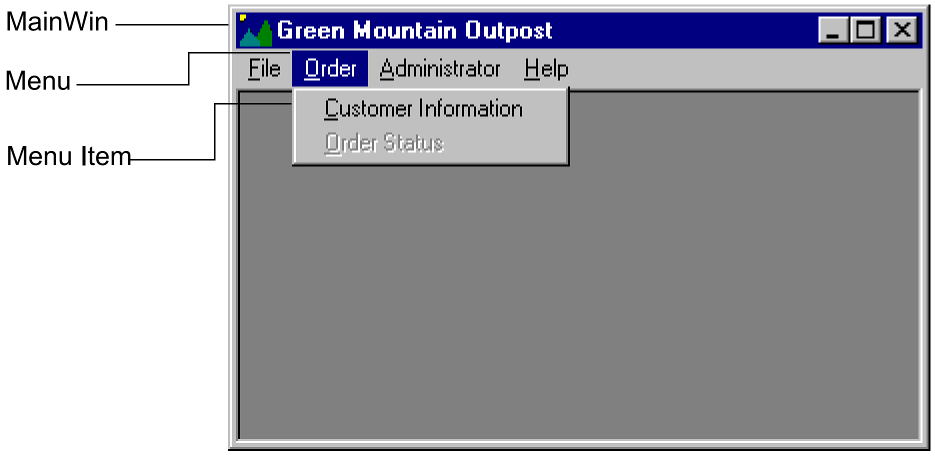
Basic object-oriented concepts

To assist you in using this tutorial the following section presents some basic object-oriented programming terms and concepts. If you are already familiar with object-oriented concepts, proceed to "The built-in recovery system" on page 12.

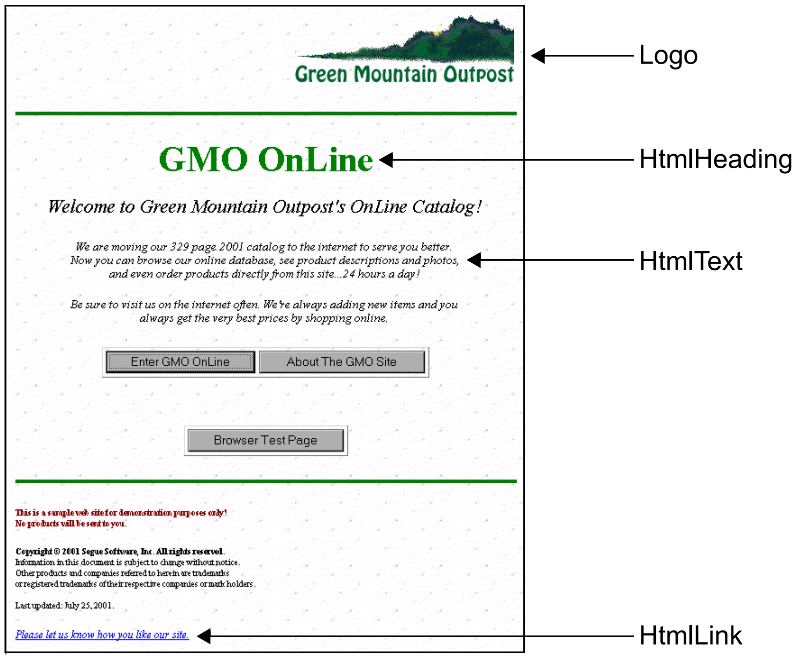
Classes

Classes are the core of object-oriented languages and are common to basic GUI architecture. A class contains a collection of information about a type of object.

The following figure shows several classes in the main window of a sample client/server application. The classes are MainWin, Menu and Menu Item.



The next figure shows several classes in the home page of a sample Web application. The classes are HtmlImage, HtmlHeading, HtmlText and HtmlLink.



GMO OnLine

Welcome to Green Mountain Outpost's OnLine Catalog!

We are moving our 329 page 2001 catalog to the internet to serve you better. Now you can browse our online database, see product descriptions and photos, and even order products directly from this site...24 hours a day!

Be sure to visit us on the internet often. We're always adding new items and you always get the very best prices by shopping online.

This is a sample web site for demonstration purposes only!
No products will be sent to you.

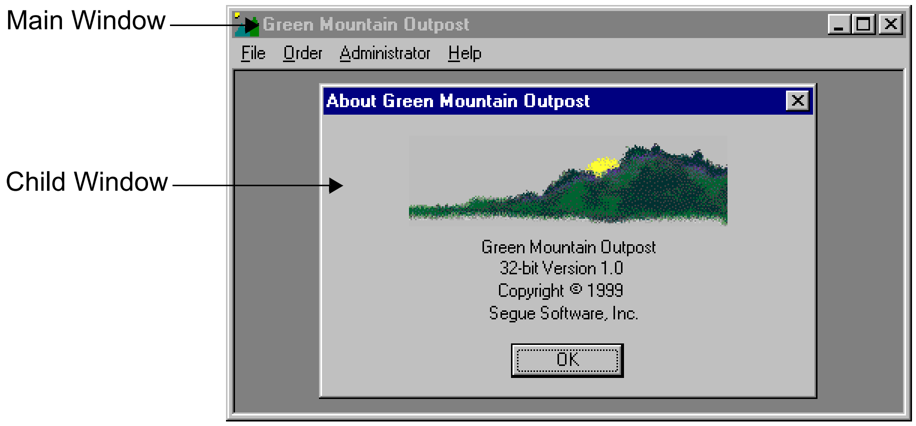
Copyright © 2001 Segue Software, Inc. All rights reserved.
Information in this document is subject to change without notice.
Other products and companies referred to herein are trademarks or registered trademarks of their respective companies or mark holders.

Last updated: July 25, 2001.

Please let us know how you like our site.

Objects An *object* is a specific instance of a class. For example, in the 4Test language, the Exit button on a window is an object of the PushButton class.

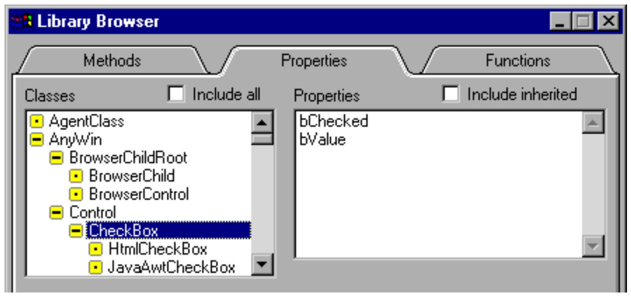
Note Objects can contain other objects. In the following figure, the About Green Mountain Outpost (GMO) window is a child window of the GMO main window.



Green Mountain Outpost
32-bit Version 1.0
Copyright © 1999
Segue Software, Inc.

Properties A *property* is the physical characteristic of an object that you can access directly. Each class has an associated set of properties, although some properties are common among classes.

The following figure shows SilkTest's Library Browser, which contains information about methods, properties and functions. What you see below are the properties defined for the CheckBox class, bChecked and bValue.



AgentClass
AnyWin
BrowserChildRoot
BrowserChild
BrowserControl
Control
CheckBox
HtmlCheckBox
JavaAwtCheckBox

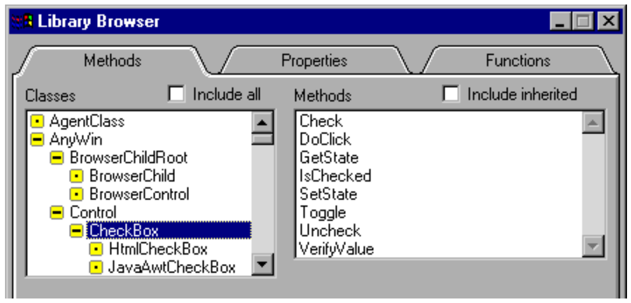
bChecked
bValue

The following table provides additional examples of properties:

Characteristic	**Property Name**
Whether a check box is checked	`State`
Whether an object is enabled	`Enabled`
The text of the selected item in a radio list	`SelText`
The index of the selected item in a radio list	`SelIndex`
The caption of a window	`Caption`
The text in the text field	`Text`
The highlighted text in a text field	`SelText`
The size and position of a window	`Rect`

Methods Actions that you perform on objects are called *methods.* The methods available to an object are inherited from the class to which the object belongs. For example, the CheckBox class defines the actions that can be performed on all the check boxes in your application such as Check, Click and Toggle. Methods defined for one class cannot be used to perform actions on a different class. For example, you cannot use methods defined for the CheckBox class to perform actions on objects in the PushButton class.

The following figure shows several methods defined for the CheckBox class:



AgentClass
AnyWin
BrowserChildRoot
BrowserChild
BrowserControl
Control
CheckBox
HtmlCheckBox
JavaAwtCheckBox

Check
DoClick
GetState
IsChecked
SetState
Toggle
Uncheck
VerifyValue

The following table provides additional examples of methods:

User Action	Method Name
You click on a push button	`Click ()`
You pick a menu item	`Pick ()`
You check a check box	`Check ()`
You clear a check box	`UnCheck ()`
You type text into a text field	`SetText ()`
You read the value of a text field	`GetText ()`
You choose an item from a list box	`Select ()`
You count the number of items in a list box	`GetItemCount()`
You close a dialog box	`Close ()`
You set the MainWin active	`SetActive ()`
You choose an item from a radio list	`Select ()`

The built-in recovery system

The built-in recovery system is one of SilkTest's most powerful features, because it allows you to run tests unattended with confidence. The failure of a single testcase shouldn't stop the rest of your tests from running. With SilkTest, when your application fails, the recovery system restores your application to a stable state, known as the *BaseState*. This allows the rest of your tests to continue even when a preceding test has failed. The recovery system will be discussed in more detail in "Understanding How the Recovery System Works" on page 110. You can also see the *online Help* for more information about the recovery system.

SilkTest Features

SilkTest offers a complete solution for the automated testing of Web and client/server applications. With SilkTest you can perform:

- Functional, operability and regression testing
- Test planning and management using the testplan editor
- Database validation using DBTester

For more information about the capabilities of SilkTest, see the *online Help*.

General application testing features

Organization of resources within a project Projects organize all the resources associated with a test set and present them visually in the Project Explorer, making it easy to see, manage, and work within your test environment.

Guidance via workflow bars The Basic Workflow guides you through the process of creating a testcase, while the Data Driven Workflow assists you in creating a data driven testcase.

Automatic completion within the Editor AutoComplete decreases the need to type text into your 4Test files by automatically completing functions, members, application states, and data types in the 4Test Editor.

Easy recording of tests The SilkTest Recorder gives you the option to create window declarations “on the fly”, and helps you set up the recovery system. By default with the DOM extension, the recorder now displays a rectangle which highlights the controls as you go through your application, as a user would. SilkTest records the appropriate verification statement in your test in the 4Test language.

True object recognition Your application is seen by SilkTest as comprised of discrete objects, with well-defined properties.

Unattended testing with built-in recovery system SilkTest intelligently handles unexpected errors and automatically returns your application to the desired state before running your next test.

Fully integrated testplan creation and maintenance You begin the automated testing process by creating *testplans*. Testplans allow you to plan and manage the testing of your application. Using criteria that you define, SilkTest allows you to run only the tests you want from within a testplan.

Database testing capabilities SilkTest enables you to manipulate your database directly using the Database Tester (DBTester). DBTester provides direct access, using SQL, from a test script to any database supported by ODBC drivers.

Full distributed testing You can test client/server applications in a networked environment.

Additional features for Web applications

The following features are available when testing Web applications:

Browser and platform independence SilkTest supports popular browsers on various platforms. You can create tests against one browser and use those tests on different browsers, enabling you to test whether your application works properly on multiple browsers. With minor modifications your tests can support different browsers and different versions of a particular browser. See the *Release Notes* for information on supported browsers.

Technology independence SilkTest works seamlessly with the different Web technologies used to develop browser-based applications.

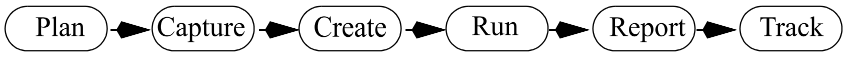
Multiple testing options using browser extensions SilkTest allows you to view and test Web applications in two different ways, depending on the browser you are using and your testing requirements. Using the Document Object Model (DOM) extension, SilkTest is able to query the browser directly for information about the objects on the Web page. Alternately, SilkTest's proprietary Virtual Object (VO) extension uses sophisticated pattern recognition techniques to identify Web page objects.

Understanding the Segue Testing Methodology

Testing is more than just creating random test scripts and running them. Testing is a skill that demands an organized, planned approach. A thoughtful approach to automated testing ensures that your testing not only uncovers a high proportion of defects but does so in a cost effective manner. Segue has developed the QA methodology to help ensure your testing is performed effectively.

The six phases of the testing process

SilkTest compliments the Segue Testing Methodology. This methodology consists of six phases:



Phase	Action	Description
I	Plan	Determine the testing strategy and define specific test requirements.
II	Capture	Classify the GUI objects in your application and build a framework for running your tests.
III	Create	Create automated, reusable tests. Use recording and/ or programming to build test scripts written in Segue's 4Test language.
IV	Run	Select specific tests and execute them against the AUT.
V	Report	Analyze test results and generate defect reports.
VI	Track	Track defects in the AUT and perform regression testing.

A summary of each phase of the Segue Testing Methodology is provided below.

Phase I: Plan

You begin the automated testing process by creating a testplan. A testplan is a document that helps you plan and manage the testing of your application. You develop testplans in SilkTest's testplan editor. Structured as a hierarchical outline, a basic testplan contains the following information:

- Descriptions of individual tests and groups of tests. You can use as many levels of descriptions as you want.
- Statements that link the test descriptions to one or more testcases that perform the actual testing of the application.
- Attributes, assigned to a specific testcase or group description. Attributes are useful for grouping tests so that you can run or report on specific parts of the testplan.

For large and complex software applications, testing every possible path is not cost or time effective; you must identify a subset of all possible tests to provide the highest coverage in the most crucial areas of the product. To get a thorough understanding of the crucial areas of your product we recommend that you review process documents such as:

- Customer Requirements
- Design Specifications
- Functional Specifications
- Marketing Requirements
- Product Documentation

Tip Segue recommends, as good business practice, that you develop a testing strategy through the development of a testplan before you create your testcases. However, developing a testplan is optional.

Phase II: Capture

In the capture phase you introduce SilkTest to the AUT by building a framework for your testing. First you record a *test frame,* which is a central repository for information about your application. A test frame stores all the data structures required to support your test scripts:

- The URL for a Web application or the command line needed to invoke the client/server application used by the SilkTest recovery system
- Constants and variables
- Descriptions, called *declarations,* for all the GUI objects in your application

Phase III: Create

After you have defined your test requirements and created a framework for testing your application, you can begin creating testcases. A testcase performs the following actions:

1. Drives the application to the state to be tested.
2. Verifies that the application's actual state matches the expected state.
3. Returns the application to its original state.

Phase IV: Run

The run phase consists of selecting one or more testcases and executing them against your application. You have the option of running individual test scripts, a collection of test scripts called a *suite,* or selecting specific testcases (using attributes) in a testplan, or the entire testplan.

Phase V: Report

During this phase you review statistical information about the running of each testcase. SilkTest automatically records the statistics in a results file when the testcase is executing. From this results file you can generate pass/fail reports.

Phase VI: Track

The final phase is to identify defects in the application you are testing. You will want to implement a defect-tracking system, to monitor your application as it progresses through development and testing to ensure that the application is performing as expected.

SilkRadar, Segue's defect tracking system, offers all the features you need to track your QA process. This tutorial does not cover the Tracking phase of the Segue Testing Methodology. For more information about SilkRadar, contact your Segue sales representative.

SilkTest File Types

There are eight types of files used by SilkTest in the automated testing process, each with a specific function.

File Type	Description	File Extension
Project	SilkTest projects organize all the resources associated with a test set and present them visually in the Project Explorer, making it easy to see, manage, and work within your test environment. The project file has a .vtp (Verify Test Project) extension and is organized as an .ini file; it stores the names and locations of files used by the project. Each project file also has an associated project initialization file: *projectname*.ini.	.vtp
Testplan	An automated testplan is an outline that organizes and enhances the testing process; references testcases and allows execution of testcases according to the testplan detail; can be of type masterplan or of subplan that is referenced by a masterplan.	.pln
Test Frame	A specific kind of include file that upon creation automatically captures a declaration of the AUT's main window including the URL of the Web application or path and executable name for client/server applications; acts as a central repository of information about the AUT; can also include declarations for other windows, as well as application states, variables and constants.	.inc

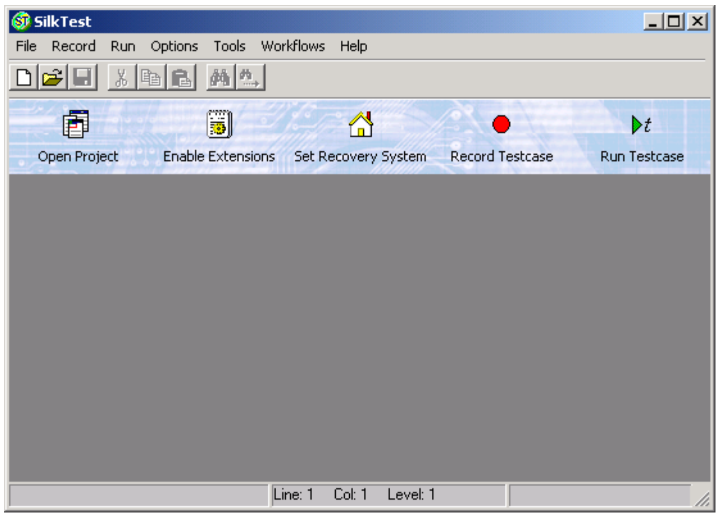
File Type	Description	File Extension
4Test Script	Contains recorded and hand-written automated testcases, written in 4Test language, that verify the behavior of the AUT.	.t
Data Driven Script	Contains data driven testcases that pull their data from databases.	.g.t
4Test Include File	A file that contains window declarations, constants, variables, classes and user-defined functions.	.inc
Suite	Allows sequential execution of several test scripts.	.s
Text File	An ASCII file; can be used to store data that is used to drive a data-driven testcase; can be used to print a file in another document (Word) or presentation (PowerPoint); can be a readme file to accompany your automation; can be used to transform a tab-delimited plan into a SilkTest plan.	.txt
Results File	Is automatically created to store a history of results for a testplan or script execution.	.res
Initialization File	Stores attribute data and query information; defaults to testplan.ini; can be modified in SilkTest's General Options dialog.	.ini

Exploring the SilkTest Window

Before you begin to create tests, you should familiarize yourself with the SilkTest main window.

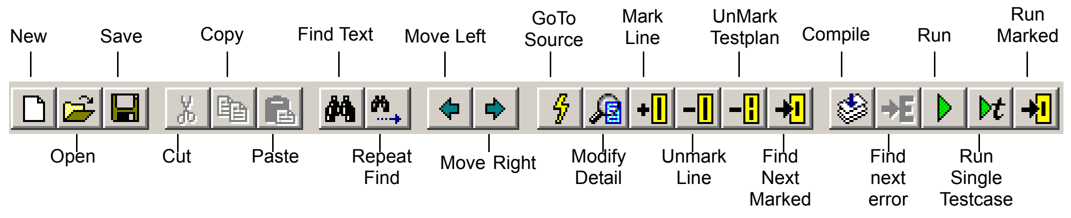
Open SilkTest

From the Start menu click **Programs/SilkTest/SilkTest**. SilkTest opens, with the Basic Workflow bar enabled. The Basic Workflow bar guides you through the process of creating a testcase. Using this workflow bar, you create a project, automatically enable and test extension settings, configure the recovery system, and record and run a testcase. If you want to turn off the Basic Workflow bar, click **Workflows/Basic**.



The standard toolbar

The *standard toolbar* provides easy access to the most commonly performed tasks. The type of file that is active determines what buttons are displayed and available on the toolbar. The image below shows the standard toolbar available for testplans.



Note You can also execute some commands using shortcut keys.

Using SilkTest with Web Applications

The challenges of Testing Web Applications

Testing Web applications poses more of a challenge to the QA engineer than testing client/server or stand-alone applications, since Web applications:

- Must run as expected on various Web browsers on different platforms.
- Can have dynamic content that may change on a weekly or even daily basis.
- Have dramatically inconsistent response times.
- Use a wide variety of technologies.

SilkTest is optimized to face these difficult challenges.

Multiple browsers

People use different browsers on different platforms to access content on the Web. Typically, you have no control over the browser or platform on which a user runs your Web application. And browsers are not the same. For example:

- Different browsers display the same pages differently: they might align and size buttons differently; they might render certain text, colors and special characters differently; they might use different heights for table rows. The differences can be dramatic.
- Different browsers have different message boxes and security dialogs.
- Different browsers support different technologies.

If you are providing your application on the Web, you might want to support multiple browsers, rather than limiting your users to those of a particular browser.

If you have made the decision to support multiple browsers, you'll want to make sure that your application works as intended and looks good on those browsers. To that end you will need a Web testing tool that is browser-independent, enabling you to see *objects* in a Web application, not just images, and you need a tool that allows you to create tests against one browser and run them against others. With SilkTest, you can do cross-browser testing.

Multiple technologies

Web applications can be implemented using a variety of technologies. These technologies are constantly evolving and new technologies are being developed all the time. SilkTest enables you to test your Web applications, regardless of how they are implemented and regardless of the technologies that comprise your application.

Dynamic content

The Web demands information to be current. To meet this demand, the content of many Web applications changes regularly, requiring your testing tool be flexible enough to keep up with the changes to your application. With SilkTest's robust scripting language, 4Test, you can quickly modify your test scripts to meet the ever-changing face of your application.

Synchronization issues

Response times in a Web application vary dramatically, depending on the speed of the server, the speed of the client, network traffic, the speed of your connection, what needs to be downloaded for a particular user, and so on. That's not a problem for SilkTest. SilkTest waits until your test page is fully loaded before it continues.

The Browser Extensions

About browser extensions

An extension is a file that serves to extend the capabilities of, or data available to, a more basic program. SilkTest provides extensions for testing applications that use non-standard controls in specific development and browser environments.

Using the **Basic Workflow**, you can automatically enable and test extension settings. You'll have a chance to do this for the GMO application in Chapter 2, "Preparing to Use this Tutorial" on page 25.

Browser extensions enable you to test Web applications. Before testing your Web application, you must specify the appropriate browser extensions on your target and host machines. SilkTest includes the following browser extensions, each of which is described briefly in this chapter:

- Document Object Model (DOM) browser extension
- Virtual Object (VO) browser extension

You must tell SilkTest which extensions to load for each AUT—regardless of whether the application will run locally or on remote machines. You do this by enabling an extension on your host machine and on each target machine before you record or run tests.

Once extensions are enabled for applications under test, SilkTest sends each Agent the list of extensions to be loaded, and the Agent dynamically loads the appropriate extensions for testing each application.

Before you can begin testing, you must determine which browser extensions to enable for the applications that you plan to test.

Document Object Model (DOM) extension

SilkTest's Document Object Model (DOM) extension makes use of the DOM standard developed by the World Wide Web Consortium (W3C). This standard, currently supported by Internet Explorer (IE) versions 5.0.*x* and higher and Netscape 6, allows external applications, such as SilkTest, to query the browser directly for information about the Web page it is currently displaying.

SilkTest's DOM extension communicates directly with the Web browser to recognize, categorize and manipulate objects on a Web page. It does this by working with the actual HTML code, rather than relying on the visual pattern recognition techniques currently employed by the Virtual Object (VO) extension. For more information about the DOM extension, refer to the *online Help*.

Virtual Object (VO) extension

SilkTest's proprietary Virtual Object (VO) extension uses sophisticated pattern recognition techniques to identify browser-rendered objects. The VO extension sees Web pages as they appear visually; it does not read or recognize HTML tags in the Web application code. Instead, the VO extension sees the objects in a Web page; for example, links, tables, images and compound controls the way that you do, regardless of the technology behind them. For more information about the VO extension, refer to the *online Help*.

What SilkTest does when you enable a browser extension

When you enable an extension on the host machine, SilkTest:

- Adds the extension's include file to the Use Files text box in the Runtime Options dialog, so that the extension's classes are available to you.
- Makes sure that the classes defined in the extension appear in the Library Browser. SilkTest does this by adding the name of the extension's help file (browser.ht) to the Help Files For Library Browser field in SilkTest General Options dialog and recompiling the help file used by the Library Browser.
- Merges the property sets defined for the extension with the default property sets.The Web-based property sets are in a file named browser.ps in the Extend directory. The file defines the following property sets: Color, Font, Values and Location. (For more information about property sets, see the *online Help*).

When you disable an extension, SilkTest reverses the preceding actions.