

**Fundamentals of CGI
Using Perl:
Academic Student Guide**



EVALUATION COPY

Fundamentals of CGI Using Perl

Developers

Ken Cohen and David Sopuch

Contributor

Brian Danks

Editor

Tom Graves

Publishers

Scott Evanskey and Joseph A. Servia

Project Managers

Dave De Ponte, Todd Hopkins and Sheila Ramirez

Trademarks

ProsoftTraining is a trademark of ProsoftTraining. All product names and services identified throughout this book are trademarks or registered trademarks of their respective companies. They are used throughout this book in editorial fashion only. No such use, or the use of any trade name, is intended to convey endorsement or other affiliation with the book. Copyrights of any screen captures in this book are the property of the software's manufacturer.

Disclaimer

ProsoftTraining makes a genuine attempt to ensure the accuracy and quality of the content described herein; however, ProsoftTraining, makes no warranty, express or implied, with respect to the quality, reliability, accuracy, or freedom from error of this document or the products it describes. ProsoftTraining makes no representation or warranty with respect to the contents hereof and specifically disclaims any implied warranties of fitness for any particular purpose. ProsoftTraining disclaims all liability for any direct, indirect, incidental or consequential, special or exemplary damages resulting from the use of the information in this document or from the use of any products described in this document. Mention of any product or organization does not constitute an endorsement by ProsoftTraining of that product or corporation. Data used in examples and labs is intended to be fictional even if actual data is used or accessed. Any resemblance to, or use of real persons or organizations should be treated as entirely coincidental. ProsoftTraining makes every effort to ensure the accuracy of URLs referenced in all its material, but cannot guarantee that all URLs will be available throughout the life of a course. When this course/disk was published, all URLs were checked for accuracy and completeness. However, due to the ever-changing nature of the Internet, some URLs may no longer be available or may have been re-directed.

Copyright Information

This training manual is copyrighted and all rights are reserved by ProsoftTraining. No part of this publication may be reproduced, transmitted, stored in a retrieval system, modified, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise without written permission of ProsoftTraining, 3001 Bee Caves Road, Austin, TX 78746.

Copyright © 2000 - 2002 by ProsoftTraining
All Rights Reserved

ISBN: 1-58143-694-7



EVALUATION COPY

Table of Contents

Course Description.....	ix
ProsoftTraining Courseware	x
Course Objectives	xii
Classroom Setup.....	xiii
System Requirements	xiii
Conventions and Graphics Used in This Book.....	xv
Lesson 1: Application Development Fundamentals	1-1
Pre-Assessment Questions	1-2
Overview of the Application Development Process	1-3
The Application Development Process.....	1-3
Platforms, Languages and Protocols.....	1-11
Client-side Versus Server-side Scripting.....	1-20
Hypertext Transfer Protocol	1-23
Lesson 1 Review	1-29
Lesson 2: Introduction to CGI and Perl	2-1
Pre-Assessment Questions	2-2
Web Architecture Overview	2-3
What Is CGI?	2-4
Why Use CGI?.....	2-7
What Is Perl?.....	2-8
Why Use Perl?	2-8
Lesson 2 Review	2-10
Lesson 3: Creating Simple Scripts.....	3-1
Pre-Assessment Questions	3-2
Key Concepts and Syntax	3-3
Scalar Variables in Perl.....	3-5
Lesson 3 Review	3-12
Lesson 4: Perl Fundamentals.....	4-1
Pre-Assessment Questions	4-2
Accessing Environment Variables	4-3
Using CGI.pm to Access Environment Variables	4-4
If Statement.....	4-6
Logical Expressions.....	4-7
Regular Expressions and Pattern Matching.....	4-8
Perl Arrays	4-9
Passing Values to Functions	4-11
Associative Arrays	4-11
Loops.....	4-13
Lesson 4 Review	4-19
Lesson 5: Perl File Input and Output Capabilities.....	5-1
Pre-Assessment Questions	5-2
File IO Using File Handles	5-3

File Modes	5-4
Using Files in Scripts	5-5
Creating a Hit Counter	5-8
Lesson 5 Review	5-12
Lesson 6: Controlling Processing and Output.....	6-1
Pre-Assessment Questions	6-2
Introduction to CGI.pm.....	6-3
Incorporating HTML into Perl Using CGI.pm	6-3
Using CGI.pm to Access Form Data	6-5
Processing User-Entered Data	6-8
Using One File to Create and Process an HTML Form.....	6-9
Using Perl to Validate Form Input.....	6-11
Testing Your Script Offline	6-12
Lesson 6 Review	6-19
Lesson 7: Saving User-supplied Data to a File.....	7-1
Pre-Assessment Questions	7-2
Saving Form Data to a File	7-3
Modifying Form Data	7-5
Pattern Matching Revisited.....	7-7
Lesson 7 Review	7-16
Lesson 8: Reading a File.....	8-1
Pre-Assessment Questions	8-2
Introduction to Data Reading.....	8-3
Pattern Matching with Regular Expressions.....	8-3
Substitution	8-5
Lesson 8 Review	8-9
Lesson 9: Introduction to Databases.....	9-1
Pre-Assessment Questions	9-2
Introduction to Database Programming.....	9-3
Four Steps to Interacting with Databases.....	9-5
Connecting to Databases.....	9-7
Structured Query Language	9-8
Quoting Revisited	9-11
Querying Table and Field Names	9-12
Web Database Query Example	9-13
Lesson 9 Review	9-21
Lesson 10: Deleting and Inserting Database Records	10-1
Pre-Assessment Questions	10-2
Modifying Data in a Database	10-3
The do Method.....	10-4
Web Database Control Example.....	10-4
Lesson 10 Review	10-17
Lesson 11: CGI Security Issues	11-1
Pre-Assessment Questions	11-2

Type of Attacks.....	11-3
Securing the CGI Script.....	11-4
Securing the Server.....	11-5
Securing Form Data.....	11-7
Securing Data Passed to Commands.....	11-9
Lesson 11 Review.....	11-19
Appendixes.....	Appendixes-1
Glossary.....	Glossary-1
Index.....	Index-1
Supplemental CD-ROM Contents.....	Supplemental CD-ROM Contents-1

List of Labs

Lab 3-1: Creating your first Perl script.....	3-9
Lab 4-1: Determining browser type and print environment variables.....	4-16
Lab 5-1: Adding a counter to myscript.pl.....	5-8
Lab 6-1: Enhancing myscript.pl to create a data entry form.....	6-12
Lab 7-1: Creating a file from data entered into a form.....	7-10
Lab 8-1: Retrieving and formatting data from a mini-database.....	8-6
Lab 9-1: Using ODBC.....	9-14
Lab 9-2: Creating a user query entry box.....	9-16
Lab 10-1: Receiving and processing data from a form.....	10-7
Lab 10-2: Deleting records and updating a Web page.....	10-12
Lab 11-1: Authenticating passwords.....	11-13

List of Figures

Figure 1-1: Web clients accessing file from server.....	1-21
Figure 1-2: Simulated code demonstrating how Web pages can use client-side scripting to appear like dynamic Web pages.....	1-21
Figure 1-3: GET request returns named resource or named process output.....	1-24
Figure 1-4: GET returns both message and header; HEAD returns header only.....	1-25
Figure 2-1: Browser, server and script interaction.....	2-3
Figure 3-1: Myscript.pl output results.....	3-10
Figure 4-1: MyScript1.pl.....	4-17
Figure 5-1: MyScript2.pl.....	5-10
Figure 5-2: MyScript.pl.....	5-10
Figure 6-1: Myscript3.pl as an interview form.....	6-15
Figure 6-2: Results of form submission for accepted candidate.....	6-16
Figure 6-3: Results of form submission for rejected candidate.....	6-16
Figure 7-1: Myscript4.pl HR interview form.....	7-13
Figure 8-1: Readdb.pl file in formatted file output.....	8-7
Figure 9-1: Database Interface module.....	9-4
Figure 9-2: Results of database query.....	9-15
Figure 9-3: Search Employee Database entry box.....	9-18
Figure 9-4: Data returned from user query.....	9-18

Figure 9-5: Search page with no results found	9-19
Figure 10-1: Employee database search and entry form.....	10-10
Figure 10-2: Results screen showing successful entry into employee database.....	10-11
Figure 10-3: Odbc.pl showing new employee entry.....	10-11
Figure 10-4: Employee Database update screen.....	10-14
Figure 10-5: Deletion results	10-15
Figure 10-6: Invalid ID results.....	10-15
Figure 11-1: Microsoft Security Advisor.....	11-7
Figure 11-2: Employee Database password screen	11-15
Figure 11-3: Invalid admin name or password result	11-16
Figure 11-4: Employee Database for authenticated user	11-16

List of Tables

Table 2-1: Virtual and actual directories for class	2-7
Table 3-1: Correct variable syntax.....	3-5
Table 3-2: Escape sequences	3-6
Table 4-1: Environment access methods	4-5
Table 4-2: Numeric and string operators	4-7
Table 5-1: File modes	5-4
Table 6-1: CGI.pm HTML-generating methods.....	6-4
Table 7-1: HTML generation methods and named parameters	7-6
Table 8-1: Special characters for pattern matching	8-4
Table 8-2: Quantifiers	8-5

Course Description

Fundamentals of CGI Using Perl teaches you how to use Common Gateway Interface (CGI), Perl programs and scripts on a Web server to carry out tasks with advanced interaction. Upon completion of this course, you will be able to implement the application development process, write print-to-screen scripts and customized Web page hit counters, create and use business-type forms that interact with text files, establish connectivity to a relational database via Open Database Connectivity (ODBC), manipulate data in a database and discuss Web server security issues related to CGI files.

Length

Fundamentals of CGI Using Perl is a 12-hour course.

Series

Fundamentals of CGI Using Perl is the first course in the CIW Application Developer series. CIW Application Developer series consists of the following two courses:

- *Fundamentals of CGI Using Perl*
- Dynamic Server Pages

Prerequisites

Students must have completed the *CIW JavaScript Fundamentals* and *Perl Fundamentals* courses or be able to demonstrate equivalent knowledge.

ProsoftTraining Courseware

This coursebook was developed for instructor-led training and will assist you during class. Along with comprehensive instructional text and objectives checklists, this coursebook provides easy-to-follow hands-on labs and a glossary of course-specific terms. It also provides Internet addresses needed to complete some labs, although due to the constantly changing nature of the Internet, some addresses may no longer be valid.

The student coursebook is organized in the following manner:

course title
table of contents
list of labs
list of figures
list of tables
lessons
lesson objectives
pre-assessment questions
narrative text
<input checked="" type="checkbox"/> graphics
<input checked="" type="checkbox"/> tables and figures
<input checked="" type="checkbox"/> warnings
<input checked="" type="checkbox"/> tech notes
labs
<input checked="" type="checkbox"/> graphics
<input checked="" type="checkbox"/> tables and figures
<input checked="" type="checkbox"/> warnings
<input checked="" type="checkbox"/> tech notes
lesson summary
lesson review
appendixes
glossary
index
supplemental CD

When you return to your home or office, you will find this coursebook to be a valuable resource for reviewing labs and applying the skills you have learned. Each lesson concludes with questions that review the material. Lesson review questions are provided as a study resource only and in no way guarantee a passing score on CIW exams.

The course is available as either an academic or a learning center version. Each of these versions has an instructor book and student books. Check your book to verify which version you have, and whether it is an instructor or student book. Following is a brief discussion of each version.

- **Academic:** Designed for students in an academic classroom environment; typically taught over a quarter (10-week) or semester (16-week) time period. Example syllabi are included on the instructor CD-ROM. The instructor's book and CD-ROM contain all answers, as well as activities (pen-and-paper-based labs), optional labs (computer-based labs), quizzes, a course assessment, and the accompanying handouts for the instructor to assign during class or as homework. No answers exist in the student book or on the student CD-ROM. Students will have to obtain answers from the instructor.
- **Learning Center:** Designed for students in a learning center classroom environment; typically taught over a one- to five-day time period (depending on the length of the course). An example implementation table is included on the instructor CD-ROM. Similar to the academic version, the instructor's book and CD-ROM contain all answers, as well as activities (pen-and-paper-based labs), optional labs (computer-based labs), quizzes, a course assessment, and the accompanying handouts for the instructor to assign during class or as homework. However, the student CD-ROM contains answers, including those to the pre-assessment questions, labs, review questions, activities, optional labs, quizzes and course assessment.

Course Objectives

After completing this course, you will be able to:

- ↗ Understand application development project fundamentals.
- ↗ Identify and work with specific server-side development environments (Perl, ASP, PHP, SSJS and JSP).
- ↗ Identify various development platforms (UNIX, Windows 2000 and Macintosh) and communication protocols (HTTP and others).
- ↗ Define CGI and Perl, and explain their significance to the World Wide Web.
- ↗ Use Perl file input and output capabilities.
- ↗ Use pattern matching and substitution to display information correctly.
- ↗ Define and use Open Database Connectivity (ODBC), and the DBI and DBD-ODBC modules.
- ↗ Discuss CGI security issues pertaining to Perl scripts.

Classroom Setup

Your instructor has probably set up the classroom computers based on the system requirements listed below. Most software configurations on your computer are identical to those on your instructor's computer. However, your instructor may use additional software to demonstrate network interaction or related technologies.

System Requirements

Hardware

The following table summarizes the hardware requirements for all courses in the CIW program. Each classroom should be equipped with enough personal computers to accommodate each student and the instructor with his or her own system.

Note: The CIW hardware requirements are similar to the lowest system requirements for Microsoft implementation (Level 1 requirements) except that CIW requires increased hard disk space (8 GB) and RAM (128 MB). This comparison may be helpful for the many training centers that implement CIW and are also CTEC because personnel at these centers are familiar with the Microsoft hardware specifications.

CIW hardware specifications	Greater than or equal to the following
Processor	Intel Pentium II (or equivalent) personal computer with processor speed greater than or equal to 300 MHz
L2 cache	256 KB
Hard disk	8-GB hard drive
RAM	At least 128 MB
CD-ROM	32X
Network interface card (NIC)	10BaseT or 100BaseTX (10 or 100 Mbps)
Sound card/speakers	Required for instructor's station, optional for student stations
Video adapter	At least 4 MB
Monitor	15-inch monitor
Network hubs	Two 10-port 10BaseT or 100BaseTX (10 or 100 Mbps) hubs
Router	Multi-homed system with three NICs (Windows NT 4.0 server)*

* Must meet universal CIW hardware requirements.

Software

The recommended software configurations for computers used to complete the exercises in this book are as follows.

- Microsoft Windows 98/Me or Windows 2000.
- Internet Explorer 5.5 or later or Netscape Navigator 4.0 or later.
- Apache Web server for Windows.
- ActiveState ActivePerl Perl Interpreter.
- Microsoft Access 2000 to view sample MDB files (optional).
- 32-bit ODBC drivers for database publishing (included with the Windows operating system).

Connectivity

Internet connectivity is required for this course. Minimum capability is achieved with a 28.8-Kbps modem; however, higher connectivity rates are recommended.

Conventions and Graphics Used in This Book

The following conventions are used in Prosoft coursebooks.

Terms	Technology terms defined in the margins are indicated in bold the first time they appear in the text. Not every word in bold is a term requiring definition.
Lab Text	Text that you enter in a lab appears in bold . Names of components that you access or change in a lab also appear in bold .
Notations	<i>Notations or comments regarding screenshots, labs or other text are indicated in italic type.</i>
Program Code or Commands	Text used in program code or operating system commands appears in the Lucida Sans Typewriter font.

The following graphics are used in Prosoft coursebooks.



Tech Notes point out exceptions or special circumstances that you may find when working with a particular procedure. Tech Notes that occur within a lab are displayed without the graphic.



Tech Tips offer special-interest information about the current subject.



Warnings alert you about cautions to observe or actions to avoid.



This graphic signals the start of a lab or other hands-on activity.



Each lesson summary includes an *Application Project*. This project is designed to provoke interest and apply the skills taught in the lesson to your daily activities.



Each lesson concludes with a summary of the skills and objectives taught in that lesson. You can use the Skills Review checklist to evaluate what you have learned.



This graphic indicates a line of code that is completed on the following line.



EVALUATION COPY

Lesson 1:

Application Development Fundamentals

Objectives

By the end of this lesson, you will be able to:

- ↻ Explain the application development process.
- ↻ Distinguish among various application development environments.
- ↻ Identify common application development platforms.
- ↻ Clarify various communication protocols.
- ↻ Determine when to use client-side or server-side scripting.

Pre-Assessment Questions

1. Which of the following lists application development activities in the order they would be performed?
 - a. Analyze the requirements, perform in-depth design, create an implementation plan
 - b. Create an implementation plan, analyze the requirements, perform top-level design
 - c. Analyze the requirements, perform top-level design, create an implementation plan
 - d. Analyze the requirements, create an implementation plan, perform top-level design

2. During which of the following application development activities should an enterprise development team decide whether or not to use prebuilt components for a project?
 - a. Creating an implementation plan
 - b. Performing top-level design
 - c. Performing in-depth design
 - d. Analyzing the requirements

3. Briefly discuss the benefits of creating pseudo-code before actually programming an application.

Overview of the Application Development Process

Application development for the World Wide Web has become a complex process, undergoing evolution similar to that of other computer software development. Many professional developers have identified certain characteristics of all projects that need to be recognized. Obvious needs include the developer's ability to understand the entire development process, to understand the various development environments available, and to use the proper tools for the task. Developers need to understand the different platforms, languages and protocols to effectively develop successful Web applications.

This lesson will introduce a few of the most popular development environments used in Web applications today, as well as some of the factors involved in selecting a development environment. This lesson will also introduce specific languages and protocols used to create Web-based applications, while giving an overview of their place in the application development arena.

Please keep in mind that all the technologies discussed in this lesson are subjects that, in their entirety, are beyond the scope of this class. These concepts and technologies are presented here to provide a basic understanding of each.

The Application Development Process

Many factors are involved in planning, developing, testing and improving successful applications. In most cases, many activities must take place before any program code is actually written. After the code is written, other activities should take place. The following list presents a general set of activities that define the life cycle of most development projects:

- Define a need for a solution.
- Analyze the requirements.
- Create an implementation plan.
- Perform top-level design.
- Perform in-depth design.
- Create the application.
- Test the system.
- Deploy the system.
- Maintain the system.

Many projects have been completed without including all the steps listed previously. However, any large project would benefit by adhering to a structured approach as indicated in the list.

This main focus of this class is the step labeled "Create the application" in the preceding list. This is the step in which the actual code for the project is written. However, remember that much more is involved in creating a successful application than writing code, as the list indicates. As each item in this list is examined, various facets of application development will be discussed.

Define a need for a solution

Before program code is written, a need for the program must exist. The first step in any application development should be a clear, concise definition of the problem that the application will address. This definition does not need to include how to fix the problem; it should simply state the problem or situation.

Ideally, the language used in the problem definition document should be that of the end user, not the programmer, unless the problem or situation is one caused by existing software. In that case, technical terms would be appropriate to describe the problem.

A discernible benefit of defining the problem before starting the solution is that the programmer will spend time fixing the right problem instead of the wrong one. Wasted development time is not easily recovered. Knowing what the problem is beforehand is immensely valuable to a development team.

Analyze the requirements

After a problem has been defined, you should determine what the application is going to do. This determination is the function of a **requirements document**, which analyzes in detail the functionality of the project. The requirements document should be user-defined, as was the problem. This arrangement frees the programmer from determining what the application needs to do, and allows him or her to focus on creating the application.

The requirements document should contain such items as the scope and core functionality of the application. Determining the requirements of an application during the coding phase is much more difficult. A detailed requirements document will help to reduce the need for changes while the application is being built. Changing what an application is supposed to do during the coding phase is also much more expensive. A well-defined requirements document can help answer questions when the coding begins, and can help guide the project through to successful completion.

requirements document

A formal definition of the specifications for software or an application.

One of the specific tasks of a requirements document is to identify all possible interactions between the end user and the application. This specification is accomplished by detailing not only the normal steps the end user might take, but also steps that might be taken if the primary sequence of events fails for some reason. All possibilities should be taken into account and planned for. For example, you could develop an application that adds employees to an employee database. The information held in the database is department, name, home phone and extension. The end user of this application can search for employees by department or by name, and can also edit or delete records from the database. In this case, the requirements document should include a normal sequence of events for a typical user interface, such as:

1. User logs on.
2. User accesses Add Employee Web page.
3. User completes Add Employee form.
4. User submits completed form for database entry.
5. System confirms results of input to user.

These steps would form a primary sequence of events for the application. As mentioned, the end user can also search the employee database, so these steps should also be detailed, as well as any other possible interfaces between the end user and the application.

The database for this application does not allow duplicate extension numbers to be added to the database. Therefore, the requirements document should detail all possible secondary sequences of events for the add employee interface, such as:

1. User logs on.
2. User accesses Add Employee Web page.
3. User completes Add Employee form.
4. User submits completed form for database entry.
5. Database traps error from duplicate extension entry.
6. System informs user of error.
7. System decides whether to abort process or send user back to add employee page for new data input.
8. If new data was entered, user submits completed form for database entry.
9. System confirms results of input to user.

If all possible scenarios of the application are detailed as previously shown, the developer will be able to create the application with increased efficiency.

Another common aspect of a requirements document is to investigate possible limitations to the application being planned. These limitations should include hardware, software and human factors engineering considerations. All limitations should be addressed and reconciled, if possible, before you start building the application.

A requirements document may also provide general terms for the design of the user interface. Typical Web applications will use some type of graphical user interface (GUI). Items that should be discussed include the type of menus to be used, icons and graphics, and the types of dialog boxes to be used for user input. The design of the user interface usually occurs in later steps of the development cycle. However, laying the groundwork for the user interface in the requirements document can help create a smooth design process later.

The requirements of an application might change during construction. Several steps can be taken to implement changes smoothly. Following are several points to remember:

- If, during the coding phase, you discover that the requirements were not defined properly, stop coding and revise the requirements document.
- Everyone involved in the project should understand the cost of changes during coding.
- Define a formal system for suggesting and implementing changes.

The related appendix contains a list of what a requirements document might include. Spend time after class examining this appendix and become familiar with the requirements it presents.

Create an implementation plan and perform top-level design

After a requirements document has been created, the next step should be to create an overall plan to implement the project. In many projects, this phase will overlap with the next – the top-level design phase. This phase can often be contained in a single document describing how the application will be built. It should contain an outline that explains the application in general terms. Each item in the requirements document should be treated as a key element and addressed in this document. This document should define how the different elements will work together, how they will communicate, and what kind of data will be passed between them. The implementation plan might also contain contingency planning for when changes occur in the project.

The implementation plan should describe the key data structures and files the application will use. If a database is to be used, its structure and data types should be defined. If the application is object-oriented, any major objects that will be used should be defined, as well as the interaction between those objects.

The user interface may have been determined by the requirements document. If not, the implementation document should outline the user interface. Other items that should be considered include input and output issues, memory management, data storage and error processing. Sophisticated applications may also need to address robustness (the ability to overcome errors), fault tolerance (methods that detect errors and help the system recover from or contain them), and performance expectations.

Another factor that should be considered in this step is whether a program already exists that can perform the job. Often, buying software is cheaper and easier than building it. Also, certain elements of the application may already exist; this option should be considered as well. Buying pre-existing components and integrating them into the application can save time and money.

In general, these steps continue to form the foundation for the actual coding. This phase of the project can help ensure a successful application, just as a strong foundation ensures the successful construction of a building.

Perform in-depth design

Many important decisions will be made during this phase. Such items as the development and server platform, the programming language, and a communications protocol must be selected. Often, many of these factors may have been pre-determined, but if not, decisions need to be made for these items. Other issues that may need to be addressed include the GUI, and any application limitations that may have been identified in the requirements document. This phase of the project can be compared to when an architect draws the final plans for a building. All the pieces need to fit together in a logical way before the first brick is laid.

One important activity that needs to take place during this phase of the project is to storyboard the application. Storyboarding will help determine the sequence of pages within the application, and optimize placement of the page's elements. The look and feel of the application will be established with storyboarding.

Considerations should include the end user's expected interface with the application, the varying sizes of windows on the end user's computer, different screen resolutions, and colors to be used. Also, the major differences among browsers and platforms should be taken into consideration. A model of the interface should be drawn on paper and tested before coding begins. Testing can include having a typical end user review the model of the interface for easy navigation, colors and presentation of content.

As mentioned, development and server platforms, programming languages and protocols need to be selected. After code creation is discussed, we will examine system testing, deployment and maintenance. The course will also present brief overviews of several of the most popular platforms, languages and protocols.

Create the application

The creation of the code for the application is the main focus of the Application Developer track. Most successful developers do not start coding without having gone through the steps already discussed. The successful developer follows a fairly strict routine when coding an application. Just as the entire process needs structure, so does the manner in which the code is written.

One of the most helpful activities that should happen before coding begins is to write pseudo-code. Writing pseudo-code means that the developer puts into common language what the code is supposed to do without worrying about all the syntactical rules of the intended programming language. When doing this step, it is important to describe the meaning of the operation itself instead of how the operation will be implemented by the programming language. Following is an example of pseudo-code:

```
Create two variables (c and r) for connection and recordset
objects
If the submitted UserID is not empty
  Assign c to a connection object
  Open the logins database
  Assign r to a recordset object that represents the result of
  a SQL query that selects everything from the login table where
  the UserIDs match
  If the recordset object is not at the end of the file
    If the submitted password matches the password from the
    recordset
      Create a Session variable equal to the UserID
      Close the connection and recordset objects
      Redirect the user to the main page
    Else
      Handle invalid passwords here
      Send the user back to the login page
    End If
  Handle invalid UserIDs here
  Send the user back to the login page
End If
Close the connection and recordset objects
End If
```

This pseudo-code does not restrict itself to any particular programming language, and could be implemented using one of several languages.

The benefits of using pseudo-code include:

- Analyzing the design details at a high level is much easier using pseudo-code. Also, changes are easier to implement at this level than at the coding level.
- Catching errors in logic is easier at a higher level. As the pseudo-code becomes distilled, errors can be caught at each level.
- Using the pseudo-code as a framework, creating the actual code becomes much easier.
- With a little editing, pseudo-code can become the comments for your routines.

After the pseudo-code for the application is written, any routines that are to be run in more than one place should be identified. A modular approach to creating the code can save many hours of actual coding. Reusing code is one of the greatest time and money savers in application development.

Before starting to code these routines or functions, check the routine (and all the pseudo-code) against the requirements document and the application architecture. Ensure that each function fits what needs to happen in the application.

When actual coding begins, it is extremely important to adhere to a methodical, predefined standard for creating the code. This standard includes such things as naming conventions, style and indentation standards, and the number and placement of comments in the code.

As mentioned, creation of code is the core of the Application Developer track. As concepts are introduced in later lessons, hands-on coding will be a part of those lessons.

Test the system

After the application has been coded, the application must be tested under varying conditions. The application should be tested before deployment. A developer with access to the code and knowledge of the code's purpose should test the application. Someone with no preconceived notion of how the application is supposed to perform should test the application from the end user's viewpoint. The application should also be tested against the different sequences of events defined in the requirements document.

The system that will host the application also needs to be tested. Often this test is done by the systems administrator, but still needs to be performed in the context of the application, because the application will reside on and affect the system in question. Like coding standards, testing is an important topic.

Deploy the system

Before an application is actually deployed, it should be tested completely on a staging server that is similar to the production server in every way except that it restricts access to the application. This staging server allows the application to be tested without a live audience. Often, testing and staging can occur at the same time; nonetheless, it is important to test the application under production-type conditions.

After the application is ready to go live, a deployment plan should be created. This plan should detail such things as:

- All components, software, HTML files, executables, graphics, sound files and miscellaneous files necessary to make the application functional.
- A plan for actual deployment, including the order in which different components of the application will be made live. This order should be determined by a component's dependence upon other components. In other words, you should not deploy HTML pages that access a database without first deploying the database on the database server.
- A deployment schedule. Issues such as access to the server, and the resetting or rebooting of the server should be addressed.
- A list of who is responsible for different aspects of the deployment.

Careful planning should go into the deployment of an application. Much hard work could be lost and excess down time could be caused by improper deployment methods.

Maintain the system

After an application is operational, you may need to adjust it to improve performance, or to arrive at the results for which the application was originally designed. This goal will probably have been achieved during the testing and staging phases, but additional work may be needed for the application to be fully functional and to perform optimally.

Many levels of maintenance must be considered. For applications with HTML interfaces, all hyperlinks must be kept up to date and verified. Content for the Web pages must be kept current, including e-mail addresses and other contact information. Corporate logos must be kept up to date. In general, time must be allotted for these activities, even if the look and feel of the application will not change very often.

In some situations, after data has been used, it is no longer needed. Then, you may need to clean up flat database files, or database tables themselves. This cleanup is often the job of the database administrator (DBA). However, communication among the developer, end user and DBA is paramount to maintaining useful, streamlined databases.

After an application is deployed, alterations may be needed so the system can meet changing needs from unforeseen situations or new requirements. In any case, the developer needs to be prepared to either adjust the application's performance, or perhaps add new modules to satisfy changing user needs.

Changes in standards or changes in versions of script engines or interpreters may also make updates necessary. Keeping abreast of the latest technologies is important for this reason. Often, new versions of software include increased capabilities or performance enhancements. Updating applications to take advantage of these changes is part of a developer's job.

The steps that have been summarized in this lesson concerning the application development life cycle are important. However, as mentioned, this course is focused on the code needed to create the application. The related appendix of this book lists some suggested resources for furthering your understanding of this subject.

Platforms, Languages and Protocols

The following is a brief overview of several of the most popular platforms, languages and protocols.

Development platforms

Obviously, several operating system platforms are available to choose from. Often, the practical choice is one that costs the least, which means using a platform that is already in place. A look at various platforms follows.

- **UNIX:** UNIX is a portable operating system written in the C programming language, and was originally developed by AT&T's Bell Labs. It is an open-source operating system. Thus, it has been developed, tested and improved through public collaboration and distributed with the source code open to all. UNIX features powerful applications, and is a multi-user, multitasking operating system. Through various programs and utilities, UNIX has networking built into its system. Through the years, various versions of UNIX have been developed, with most adhering to the standards of the UNIX community. Linux is a UNIX-type platform that has become popular and has improved the interface for the operating system. Different versions of Linux have also appeared, such as Red Hat, Caldera OpenLinux and VA Linux, all adding a different twist to the UNIX-type operating system. Sun

Microsystems' Solaris is another UNIX system that has added to the popularity of UNIX-style computing.

The standards mentioned previously are what make UNIX an attractive choice for the developer. The open source allows applications developed for one type of UNIX to run on any type that adheres to the standards.

Applications can be developed in a number of programming languages when using UNIX. C, C++, Java, Perl and Tcl are all supported within the UNIX environment. CGI scripts written in a variety of languages can be run using UNIX-compatible Web servers such as Apache, NCSA and CERN (or W3C HTTPD.)

Many UNIX developers are dedicated to UNIX editors such as emacs (the name was derived from Editing MACroS), and avoid using an integrated development environment (IDE). However, many IDEs are supported in the UNIX environment. Simple HTML editors, as well as complex tools for C, C++ and Java are available for application development on the UNIX platform.

- **Microsoft Windows NT/Windows 2000:** Windows NT/Windows 2000 servers are Microsoft networking products. Windows NT 4 features a user interface similar to that of Windows 95. Windows NT is portable and capable of multitasking as well as multithreading; it also features multiprocessor support. The operating system is written in the C programming language, which contributes to its portability.

As with UNIX, applications can be written in a variety of programming languages. Programs written in C, C++ and Java are all supported. Internet Information Server (IIS) is Microsoft's Web server, and is available on the NT 4 Option Pack. CGI programs written in a variety of languages are supported by IIS. Also, IIS 4 and higher natively support Microsoft Active Server Pages. Windows NT/2000 also supports other Web servers such as the Windows version of Apache.

Windows NT/2000 support an extensive range of IDEs for application development. Everything from HTML editors to programs for C, C++ and Java are widely available for use on Windows NT/2000 operating systems.

- **Mac OS:** Mac OS is Apple's operating system for its Macintosh and iMac computers. It is a multitasking, multiprocessing operating system, and is especially useful to many graphic designers and online visual artists.

The Mac OS can be successfully used as an application development environment. Many IDEs exist for the Macintosh, including those for C, C++ and Java development.

Server-side development technologies

This section will present a brief overview of several server-side programming language environments.

- **Java:** Java is an object-oriented, multithreaded, robust and secure programming language developed by Sun Microsystems. It was designed to resemble C++, but it simpler to use and is designed for use over distributed networks. Java creates mini-applications for the Web, called applets, and stand-alone applications. Java programs are portable because they are compiled into byte code and executed by a Java Virtual Machine (JVM). A JVM is software that acts as an interface between the compiled code and the actual processor that performs the program's instructions. JVMs exist for all major operating systems; therefore, code written for one platform can be easily transferred to another as long as the JVM is present.

Java is object-oriented. This characteristic allows developers to focus on built-in and custom objects and their attendant methods and properties instead of thinking strictly in terms of procedures. In Java, a class is a collection of data, with methods that act on that data. Classes are arranged in a hierarchy, and a subclass can inherit properties and methods from its superclass.

Java is a secure language, which is especially important given its distributed nature. Byte-code verification and the sandbox model are two features of Java's security. The Java interpreter performs byte-code verification on any untrusted code it loads. The sandbox model means that untrusted code has severe restrictions on what it can and cannot do.

Though the syntax and control structures used in Java are fairly straightforward, learning to use the many classes and libraries available with Java can be complex. Java is a comparatively difficult language for new programmers to learn.

Several tools are available to aid in the creation of Java programs, including JBuilder, VisualAge for Java and Visual Café.

- **C++:** C++ is an object-oriented programming language used to create large-scale application programs. C++ is a superset of the C programming language and is designed to make C a safer programming environment. C++ implements the object-oriented paradigm, including classes, inheritance and polymorphism. The C++ environment features libraries to aid in the creation of applications. C++ can be used on a variety of computers ranging from PCs to supercomputers. The American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) have standardized C++. This standardization helps to make C++ programs portable, as vendors create standardized compilers for various platforms.

C++ has the speed of the C language, and is highly modular, so code can be reused or easily modified. Not only is C++ suited for formulas and computation; it is also suitable for use in graphics-intensive applications, database applications, and all types of business application programs.

Like Java, the syntax of C and C++ programming is easy to learn, but learning the libraries and classes available in C++ is challenging for new programmers.

Many tools exist for creating applications in C++. These include C-Forge, CMeister, Code Crusader and the Borland C++ IDE.

- **Active Server Pages:** Microsoft's Active Server Pages (ASP) technology provides a development environment for building dynamic, interactive Internet and intranet applications. Server-side scripts written in languages such as Visual Basic, Microsoft JScript or VBScript, Java or C are used to implement ASP applications. These scripts are embedded into HTML pages and executed on the server with the resulting HTML returned to the client. The use of an existing scripting language is one of the advantages of ASP. This feature allows a developer who is already familiar with a language to quickly migrate to the ASP development environment.

ASP applications are client-independent. The same application will perform on either of the major browsers because ASP applications return only HTML, and do not depend on the client for any processing of ASP code. ASP applications can interact with ODBC-compliant databases on the Web server. These databases can include Microsoft Access or SQL Server, Oracle, Informix or Sybase. One of the most powerful aspects of ASP is its ability to create state and manage Web sessions.

The ASP environment includes built-in objects and components that enhance the functionality of the application. Components are reusable program building blocks that can be combined with other components in an application. They expose properties and methods to be manipulated by the developer. Some components are prepackaged with ASP, and hundreds of third-party components are available. The built-in objects mentioned previously offer a wide variety of functionality, including information about the server environment, information about the HTTP request and response, and application and session management objects.

When ASP was first released in 1996, Microsoft Internet Information Server (IIS) for Windows NT and Personal Web Server for Windows 95/98 were the only server environments that supported it. In 1997, ChiliSoft developed Chili!ASP, a software product that supports ASP functionality on Web servers other than IIS. Now platforms such as AIX, HP-UX, Linux and Solaris can host ASP applications. Also, Halcyon Software has created Instant ASP, software that allows developers to deploy ASP applications to any Web server or OS platform.

ASP applications can be created with almost any editor. However, the best environment is Microsoft Visual Studio, which includes Visual InterDev. Visual InterDev is a complete IDE for developing ASP applications.

Later lessons in this track will introduce this server-side application development environment.

- **CGI/Perl:** The Common Gateway Interface (CGI) is a standard way for a Web client to invoke an application program residing on a Web server. After the program executes, CGI sends the results of the program back to the client. CGI uses Transmission Control Protocol/Internet Protocol (TCP/IP) and Hypertext Transfer Protocol (HTTP). CGI programs can be written in a variety of programming languages, including C, C++, Java and Perl. Perl is probably the *de facto* standard for CGI programs.

Perl is an acronym for Practical Extraction and Report Language. It was originally written by Larry Wall and has been enhanced and extended by thousands of programmers. It is a scripting language similar in syntax to C and includes many powerful commands that make processing text files easy. HTML files and form data posted to a Web server are text, so Perl is a good choice for processing that text. Perl is generally easier to learn than C++ or Java, but can also be used to create complex programs. Also attractive is the large number of working Perl scripts widely available over the Internet. Also, modules exist that can be included in Perl programs to automate many tasks. Perl programs can integrate easily with all major databases through the use of these modules.

Programs written in Perl for CGI implementation can run on a variety of platforms, usually with little or no alteration from one platform to the next.

IDEs exist for the Perl development environment. These include Perl Builder, Perl Scribble and synEdit. The Perl Development Kit (PDK) offers a visual debugger and utilities for creating enhancements to Perl programs.

This book covers CGI using Perl. Upcoming lessons will introduce practices and concepts particular to CGI and Perl.

- **PHP:** PHP Hypertext Preprocessor (PHP) is a scripting language and interpreter similar to Microsoft Active Server Pages in functionality. However, PHP is part of the open source movement, and is used mostly on the Linux platform with the Apache Web server. Like ASP, PHP scripts are embedded into HTML pages, with the scripts interpreted by the PHP processor, and the resulting HTML sent to the client. This arrangement eliminates the browser-support issue just as with ASP. PHP is platform-independent, and versions exist for many Web servers, including IIS.

PHP was conceived by Rasmus Lerdorf in 1994 and was quickly embraced by the open-source community. As with many open source projects, PHP has evolved very quickly due to the number of its contributors. As in Perl, many modules exist as extensions to the PHP interpreter. These modules automate many tasks in the PHP environment and help create sophisticated, dynamic, database-driven applications for the Web.

PHP uses syntax similar to Perl and C. As with Perl, many text-handling functions are included in the language, making it a powerful tool for parsing Web form data. PHP applications can interface with a number of databases including MySQL, mSQL, PostgreSQL, Sybase, Oracle and even ODBC-compliant databases.

IDEs for PHP include K PHP Develop, CoffeCup, Edit+, Homesite, Multi-Edit and UltraEdit.

- **SSJS:** Server-Side JavaScript (SSJS) is Netscape's response to ASP. Like ASP and PHP, SSJS is embedded script in an HTML page. The server executes the script and returns HTML to the client. SSJS applications are created using JavaScript and a combination of objects that exist on the server. The hierarchy and functionality of these objects is similar to the built-in objects mentioned in the previous ASP synopsis. The fact that JavaScript is used is an advantage, as JavaScript is the standard language of the Web.

One minor disadvantage for SSJS programs is that they must be compiled before being run. A major disadvantage is that SSJS is supported only on Netscape Web servers. Netscape is behind in its proportion of servers on the Web when compared to Apache and IIS. This lag has probably limited the number of applications built using SSJS.

IDEs listed on the Netscape Web site include IntraBuilder, DreamWeaver UltraDev and iNet Developer. Though most of these tools are intended for Java development, SSJS applications can be developed using them.

- **JSP and Java Servlets:** These two server-side technologies use Java as their foundation. Java Server Pages (JSP) are similar to ASP and PHP in that a JSP page contains HTML, as well as embedded Java code and JavaBean components. The embedded code and components are executed on the server with the results sent to the client in the form of HTML. The code is executed via a Java servlet. This is a server-side program that is invoked via an HTTP request and returns its results as an HTTP response. A servlet can be compared to CGI programs, but is more accurately likened to a Java applet that runs on the server. The life cycle of a servlet is like that of an applet, and it runs inside a Java Virtual Machine (JVM).

Because these two technologies use the Java language, the learning curve for new programmers is probably steeper than those for similar technologies such as SSJS, ASP and PHP.

Several of the IDEs mentioned for SSJS and Java can be successfully used to create JSP applications and Java servlets.

Client-side development technologies

- **JavaScript:** JavaScript was developed by Netscape to address the shortcomings of HTML used within client browsers. HTML can respond to user input only in a limited way. It cannot validate data, or manipulate the objects within the browser, but JavaScript can do those activities and others. JavaScript allows developers to add interactivity to Web pages on the client's machine. JavaScript is the standard language for client-side scripting, as most browsers support it in one form or another.

JavaScript is a scripting language. Script engines are built into the browsers and are responsible for interpreting the commands embedded into the HTML document. These commands are in the form of built-in language functions or methods, user-defined functions, or in-line scripting embedded in HTML tags. JavaScript's syntax is similar to that of C or Java, but its learning curve is somewhat shorter because JavaScript has no classes of objects as Java does. JavaScript is referred to as an object-based language because it manipulates objects that exist outside the language.

Many IDEs exist for creating JavaScript code. Some of these are ScriptBuilder, Infuse, Homesite and PowerSite.

- **VBScript:** VBScript is a Microsoft scripting language intended for the same basic purposes as JavaScript. VBScript is a subset of Visual Basic for Applications (VBA), which is itself a subset of Visual Basic (VB). VBScript's syntax is inherited from its parent language.

VBScript is used in the same manner as JavaScript: It is embedded within an HTML document and interpreted by a script engine. The language provides built-in functions or methods and the developer can create user-defined subroutines and functions. VBScript also supports in-line scripting. Like JavaScript, VBScript is considered to be object-based.

The Netscape browser does not support VBScript. For this reason, VBScript is usually found only in server-side applications, or on the client side in intranet applications intended for the Internet Explorer browser.

IDEs for VBScript include Microsoft Visual InterDev, Homesite and VisualBasic.

Refer to the related appendix for information concerning recommended reference material for the languages described in this section.

Communication protocols

Following is a brief overview of communication protocols.

- **COM/DCOM:** Microsoft's Component Object Model (COM) and Distributed Component Object Model (DCOM) allow developers to organize the functionality of an application into logical units, or objects. COM provides the framework for developing and supporting high-level software services such as Object Linking and Embedding (OLE). COM also provides the underlying services of interface transactions, the object's life cycle management, licensing, and event services (implementing one object as the result of an event that has occurred to another object). Other major services include security, database access, data transfer, asynchronous communications and automated invocation of components. COM also offers the flexibility of implementing COM objects via multiple network transports, such as TCP, User Datagram Protocol (UDP), Internetwork Packet Exchange (IPX), Sequenced Packet Exchange (SPX), HTTP, or as a queued process. COM is supported on multiple platforms.

In essence, COM allows components that comply with the COM specification to interact with each other no matter what language the component was written in. As such, COM components are both tool-independent and language-independent. So components built in Java, Visual Basic and C++ can work together as long as they conform to the COM specification.

Major advantages of creating applications that use the COM specification are the reusability of objects, the freedom to use different languages and tools, and the ability to use off-the-shelf components. COM services are implemented in a standard way, no matter where the COM object is located. The COM specification is mature, well documented, and in common use.

DCOM allows objects in client programs to request services from server program objects across networks. A Web page could be processed not on the Web site server, but on another specialized server on the network. The Web site server, acting as a client, and using DCOM interfaces, can forward a Remote Procedure Call (RPC) to the specialized server object. The specialized server object performs any necessary processing and returns the results to the Web server, which in turn delivers the results to the Web client. DCOM is based on the COM specification, and can connect applications to servers running COM objects.

- **CORBA:** Common Object Request Broker Architecture (CORBA) is a specification for managing program objects across distributed networks. A key element in CORBA is the Object Request Broker (ORB). An ORB makes it possible for client objects to make server requests without having to know where in a network the server object is located, and exactly what the interface to the server program looks like. The ORB can be considered middleware that acts as the "broker" between a client request for a service from a distributed object and the completion of that request. To make requests or return replies between the ORBs, the General Inter-ORB Protocol (GIOP) is used and, for the Internet, the Internet Inter-ORB Protocol (IIOP) is used. IIOP maps GIOP requests and replies to the Internet's Transmission Control Protocol (TCP) layer in each computer involved in a transaction.

CORBA is similar to DCOM. Program objects written in any language can communicate with other program objects across a network as long as the objects comply with the CORBA specification.

The Object Management Group (OMG) developed CORBA. This group of vendors was formed in 1989 with the goal of creating a standard architecture for communication between objects in distributed networks. Microsoft and OMG have tentatively agreed to create gateways that will allow client objects developed in COM to be able to communicate with CORBA servers. Also, these gateways would allow client objects developed in CORBA to communicate with COM servers.

- **JavaBeans:** JavaBeans from Sun Microsystems is an object-oriented programming interface that builds reusable components that can be deployed in a network on any major platform. Like Java applets, JavaBeans components, also referred to as Beans, can give Web pages (or other applications) interactive capabilities or supply dynamic content.

When JavaBeans are part of an application, the properties of the JavaBean (its physical characteristics) and methods (the things it can do) are visible to other JavaBeans. This visibility allows them to learn each other's properties and methods dynamically and to interact according to the program's needs.

JavaBeans are developed with the Beans Development Kit (BDK) from Sun and can be run on any major platform such as Windows 95/98, Windows NT, UNIX, or Macintosh, or inside a number of application environments such as browsers and word processors. To build JavaBean components, developers write language statements using Java and include JavaBeans statements that describe component properties, methods and events that trigger a bean to communicate with other beans in the same container or elsewhere on the network.

JavaBeans also have persistence, which is a mechanism for storing the state of the component. Persistence allows JavaBeans to remember data connected with a particular session, and for that data to be reinstated at a later session.

JavaBeans may be used as embedded objects or components within applications such as Microsoft Office, Internet Explorer and Visual Basic. Sun provides a Java plug-in that acts as a bridge and provides a way to package and register JavaBeans as COM components.

Enterprise JavaBeans are those that will be shared across distributed networks. These JavaBeans run in a special environment called an Enterprise JavaBean (EJB) container. The EJB container hosts and manages the JavaBean in the same way that a Java Web server hosts a servlet or an HTML browser hosts a Java applet. Enterprise JavaBeans cannot function outside of the EJB container. The EJB container manages every facet of the JavaBean at runtime, including remote access to the JavaBean, security, persistence, transactions, concurrency and access to resources. Most EJB servers support either the Java Remote Method Protocol (JRMP) or CORBA's Internet Inter-ORB Protocol (IIOP). Use of these protocols allows applications to invoke programs that use JavaBean technology.

COM/DCOM, CORBA and JavaBeans are beyond the scope of this class. These definitions have been provided as part of a general overview of the application development process.

HTTP is more specific to the technologies covered in this course. However, before HTTP is reviewed, this lesson will present a discussion of client-side and server-side scripting.

Client-side Versus Server-side Scripting

The distinction between a client and a server is that the client makes a request, and the server responds to the request. The Internet (Web) client/server model has a number of servers as distinct entities on the network. The client finds the server using a Uniform Resource Locator (URL) as an address, then connects to the server by issuing a request. The request defines the interaction that the client will have with the server. The base case (non-dynamic interaction) is that a user is requesting a static file. The request for the file is processed and displayed on the client side via a response from the server. Any script commands that may be embedded in the file are executed on the client side. The major advantage in this scenario is that the server is not concerned with the type of client requesting the file. For example, any client that can use HTML can use a file from a Web server.

The server can respond to many requests very quickly with this model, because very little per-client processing is required. This activity is illustrated in Figure 1-1.

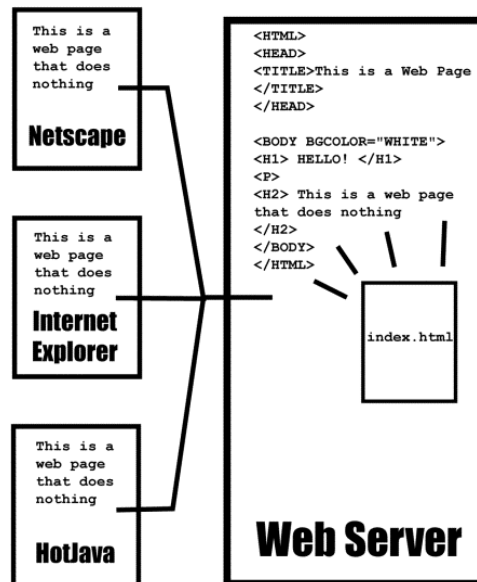


Figure 1-1: Web clients accessing file from server

With the numerous capabilities offered by HTML, Dynamic HTML (DHTML), JavaScript and Java, it may not be apparent why more capabilities should be added at the server end. When a client requests a Web page that contains embedded active content, these client-side technologies perform operations after the page is downloaded, thereby distributing the work. This distribution ultimately speeds up the Web site, because the server is not bogged down with calculations for each client. The time saved allows more clients to be serviced.

An important part of optimizing Internet performance is to distribute as much of the work as possible. See Figure 1-2.

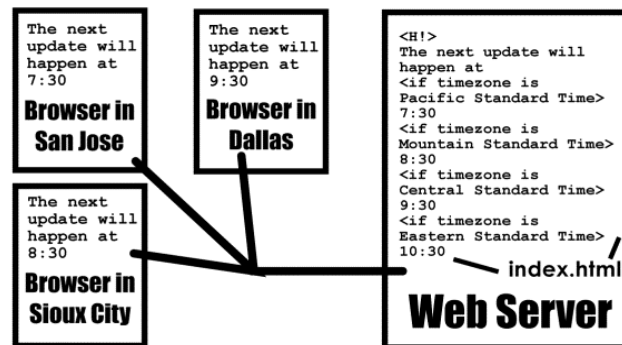


Figure 1-2: Simulated code demonstrating how Web pages can use client-side scripting to appear like dynamic Web pages

In many cases, such as the one depicted in Figure 1- 2, distributing the work to clients is fastest and easiest, letting each client decide which data should be displayed. Pages that will be vastly different for each client, however, should probably be created through server-side scripting. Also, overuse of client-side scripting can result in very large files that can take a long time to download.

This model has other shortcomings. A dynamic Web page might be built from data that is managed by another application. The client would need special drivers to access that data, meaning every visitor to the Web site would have to download extra driver files. Another important part of optimizing Internet performance is to minimize the number of network transactions. Downloading one large file is faster than creating many smaller connections, because every TCP/IP connection has an overhead cost. Clients should not connect to data sources individually.

Besides the issues of performance, security risks are inherent in giving every client direct access to data sources. The data source often cannot manage its own security. Giving different clients variable degrees of access to data makes the issue even more difficult to resolve from the client side. If the client has any kind of third-party contact, security is virtually impossible for the server to manage. The advantages of using server-side scripting should be apparent based on this discussion. The use of back-end data sources allows pages to be far more dynamic than with only client-side scripting.

The optimal Web page combines server-side and client-side scripting. The general rules are as follows.

- Any access to data that resides on the client, such as the time or the type of browser, should be implemented with client-side scripting.
- Any access to data that resides anywhere except the client should be implemented with server-side scripting.
- Minor changes to HTML layout and properties should be managed on the client side.
- Major differences in the HTML or media that a client will receive should be managed on the server.

Following is a brief overview of HTTP.

Hypertext Transfer Protocol

HTTP is extremely important because it is used for messages between clients and server applications. To illustrate HTTP, several transactions will be examined in detail.

Passing information with HTTP

When an HTML form is submitted to a server using either the GET or POST method, each field in that form should have a name. HTTP will use this name to track the values in the form. For example, for a text field called *text1* containing the word *Alex*, that value would be encoded into a request as:

```
Text1=Alex
```

If the request contains more than one value, the values are separated by an ampersand (&). No spaces are allowed.

```
Text1=Alex&Text2=Crenshaw&Text3=Hello
```

The following examples show how names are used with GET and POST.

HTTP methods

The three methods of HTTP 1.0 are defined in RFC 1945 (www.faqs.org/rfcs/rfc1945.html). The methods are GET, HEAD and POST. These methods, and the data that accompanies them, are the most important parts of HTTP, at least to the server script developer.

GET

The GET method is used to retrieve a resource, such as a Web page. The URL at the top of a browser is sent as a GET request to a Web server. The syntax is the GET command followed by the requested resource, in URL format. The first part of the URL resolves the server that should be used. The rest of the URL formulates the GET request. The server can respond with a file or send the output of a data-producing process, which is the whole point of server-side scripting.

The syntax of the GET method, in its simplest form, is the keyword GET followed by the document address:

```
GET /index.html
```

The client can send more information, however, to help the server process the request more completely. First, the client can send the HTTP version number. On subsequent lines, additional information can be sent in the form of a header. For example, the client could inform the server that the client's browser is Netscape Navigator 4.07 and the operating system is Windows NT.

```
GET /index.html HTTP/1.0                <-Request, with version
User-Agent: Mozilla/4.07[en](WinNT; I) <-Header Information
```

The GET request can access normal files or processes (programs) that reside on the server. This activity is illustrated in Figure 1-3.

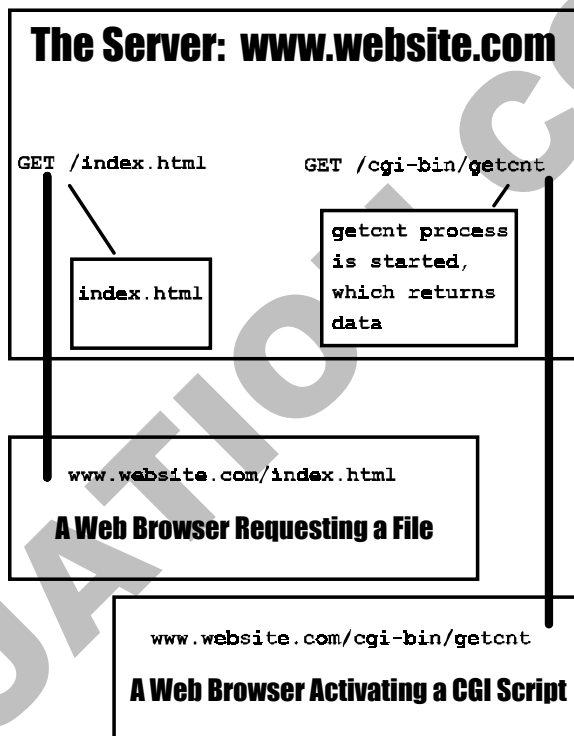


Figure 1-3: GET request returns named resource or named process output

When the GET method is being used to invoke a server process like parsing form data, additional information (parameters) are usually sent. The basic encoding is shown in the previous section entitled "Passing information with HTTP." The GET method attaches this extra information to the end of the request, separated by a question mark. The following is an example of how a Perl script called `message.pl` would be invoked from an HTML form with the same values used in the previous example:

```
GET /cgi-bin/message.pl?Text1=Alex&Text2=Crenshaw&Text3=Hello
```

HEAD

The HEAD method is similar to the GET method, with identical syntax. The difference is that the HEAD method returns only the header instead of the whole resource. The header is information about the rest of the file. It can tell you whether a new version of the file should be downloaded, display the file size without downloading, or provide a list of all the files in a directory. HEAD is shown in Figure 1- 4.

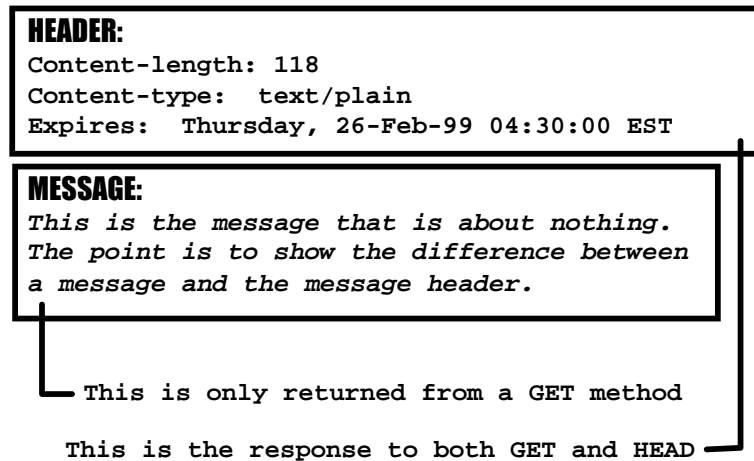


Figure 1-4: GET returns both message and header; HEAD returns header only

POST

The POST method sends information to the server. Whereas the GET command retrieves data, the POST command submits data. The difference between the two is minor when you consider the invocation of a server process (for example, CGI). A program can be called with either GET or POST, and both can send data as parameters to a program. A primary advantage of POST over GET is that POST is not limited by size. With some servers, the GET method is allowed only a certain number of characters in the URL, and that number depends on the particular server. POST does not embed the request in the URL; it uses standard input.

The actual use of POST is similar to that of GET in syntax. The POST keyword is similarly followed by the target's address (URL). POST can also be followed with header information. The difference is that POST then includes a message. The header information with a POST request can contain the same information about the client, such as the browser name and version, but the header also contains information about the body of the POST request itself.

The body of the POST request contains the parameters being sent to the program, using the encoding discussed in the previous section entitled "Passing information with HTTP." Following is a sample POST request:

```
POST /cgi-bin/message.pl
Content-type: application/x-www-form-urlencoded
Content-length: 37
```

```
Text1=Alex&Text2=Crenshaw&Text3=Hello
```

As mentioned, GET and POST are interchangeable in most situations. Generally, GET should be used to retrieve information. POST should be used if the request sends information that is to be parsed or that will modify data stored on the server.

Lesson Summary



Application project

One way to learn about Web server-side development is to investigate the many languages and tools available for creating dynamic, data-driven Web applications. Many resources exist that provide valuable information about various languages and tools. For this application project, investigate some or all of the following CGI and Perl Web resources.

CGI/Perl resources:

- www.perl.com/
- <http://cgi.resourceindex.com/>
- www.cgidir.com/
- <http://perlmasters.com>
- www.activestate.com/

Open source development:

- <http://devshed.com>
- www.openresources.com/
- www.opensource.org/
- www.osdn.com/
- <http://sourceforge.net>

Development tools:

- www.xarka.com/optiperl/index.html
- www.solutionsoft.com/perl.htm
- www.activestate.com/Products/Komodo/
- www.ultraedit.com/products/index.html
- www.perl.com/cs/user/query/q/6?id_topic=41



Skills review

This lesson presented a brief overview of the application development process. The key steps involved in planning and deploying Web applications were covered, including analyzing and defining the requirements of an application; as well as designing, creating, testing, deploying and maintaining the application. This lesson also presented a brief overview of several common platforms, languages and protocols that can be used in application development. The lesson covered the difference between client-side and server-side scripting. Finally, an overview of the Hypertext Transfer Protocol was presented.

Now that you have completed this lesson, you should be able to:

- ✓ Explain the application development process.
 - ✓ Distinguish among various application development environments.
 - ✓ Identify common application development platforms.
 - ✓ Clarify various communication protocols.
 - ✓ Determine when to use client-side or server-side scripting.
-

Lesson 1 Review

1. What is a major benefit of a requirements document?

2. What is pseudo-code and why is it helpful?

3. What are some IDEs for the Perl development environment?

4. What is one disadvantage of client-side scripting?

5. State the combination of rules that should be used to determine when server-side scripting should be used instead of client-side scripting.
