Perl Programming and CGI Scripting

Students Will Learn:

- Language Syntax
- Pattern Matching with Regular Expressions
- Flow Control
- Arrays, Hashes and Complex Data Structures
- File and Directory I/O
- Classes, Objects and Methods
- Debugging

- Procedural and Object Oriented Programming
- Command Line Interaction
- Subroutines and Scope
- CGI Scripting using CGI.pm
- Forms Processing
- Database Access (DBI)

Course Description: This hands on Perl programming course provides a thorough introduction to the Perl programming language, teaching attendees how to develop and maintain portable scripts useful for system management, data manipulation, and Web CGI programming. Emphasis is placed on built-in subroutines that can be used to help conveniently build fast, portable and efficient scripts. Extensive hands on exercises provide practice in report creation, pattern matching, string manipulation, file I/O, command line processing, and debugging. Students are shown how to extend Perl's basic functionality with packages and loadable modules. The final day encompasses CGI scripting with Perl as well as database access using the DBI module. Attendees are shown how to validate form data, how to perform robust database access, and how to generate HTML output in order to create a dynamic web site. Attendees who do not need or want in-depth coverage of CGI scripting should attend HOTT's *Perl Scripting* course instead of this course.

Course Prerequisites: Prior scripting experience or knowledge of fundamental programming concepts. For CGI programming, knowledge of HTML fundamentals and SQL is helpful but not required.

Perl Programming and CGI Scripting Course Overview:

Introduction to Perl

- Origin and Design Goals of Perl
- Overview of Perl Features
- Getting and Installing Perl
- Accessing Documentation via perldoc
- HTML-Format Reference Documentation
- Perl Strengths and Limitations

Using Variables

- Scalar Variables
- Introduction to Standard Data Types
- Retrieving Standard Input Using the Default Variable $_
- Reserved Scalar Variables
- Assigning Strings and Numbers to Scalar Variables
- Declaring Constants for Persistent Values
- Using strict to Declare Variables

Getting Started With Perl

- Explicit Invocation of the Perl Interpreter
    - Running Perl on UNIX vs. Windows
    - Running Perl from the Command Line
    - Using Command Line Options
    - Using Debug Mode
- Implicit Invocation of the Perl Interpreter
- Running and Debugging Perl Scripts
- Simple and Compound Statements
- Fundamental Input Techniques
- Using the print Function to Generate Standard Output

Pattern Matching in Perl

- Regular Expressions in Perl
- Using Pattern Matching Operators
- Altering Data with Substitutions in Regular Expressions
- Using Backreferences to Capture Data from Regular Expression Matching
- Global and Case-Insensitive Matches
- Altering Data with Character Translation
- Using Variables in Patterns

Operators

- Introduction to Fundamental Operators
- Operator Precedence and Associativity
- Using the Ternary Operator ?: as a Shortcut for the if Statement
- Using <FILEHANDLE> and <> File I/O Operators for Standard Input/Output
- Using the Shortcut Operators +=, -=, *=, /=

Flow Control: Conditional Statements and Looping

- Conditional Expressions and Logical Operators
- if/else/elsif and unless
- Constructing switch/case Equivalent Expressions
- while Loops and do Loops
- for and foreach Loops
- Labels
- Altering Program Flow with next, last, and redo
- Trapping Errors with the eval Function
- Terminating a Script with exit

Arrays and Hashes

- Defining Numeric Index Arrays
- Defining Associative Arrays
- Sorting Arrays with the sort Function
- Adding and Deleting Items Using push, pop, shift, and unshift
- Using slice, splice, and reverse
- Other Array Manipulation Techniques
- Looping through an Array
- Merging Arrays
- Associative Array Manipulation Functions
- Introduction to Hashes
- Preallocating Memory to Optimize Hash Performance

String Manipulation

- String Comparison
- String Relations
- Concatenation
-
- Substring Manipulation
- Using chomp and chop to Eliminate EOL Characters
- Escape Characters for Formatting
- String Manipulation Functions

Subroutines and Parameters

- Simplifying Scripts with Subroutines
- Defining and Calling a Subroutine
- Passing Arguments by Value
- Passing Arguments by Reference
- Using return to Return a Value
- Controlling Variable Scope using my and local Keywords

Packages and Modules

- The Power of Packages and Modules
- Introduction to Standard Modules
- Where to Find Modules on the Internet
- Installing a Module on UNIX or Windows
- Creating Packages for Portability
- Using Packages to Create Isolated Namespaces and to Separate Code
- Creating Modules
- Creating and Using Symbols in a Module
- Using the Exporter to Export Symbols from a Perl Module

File and Directory I/O

- Using open and close
- File Open Modes
- Reading Files into Arrays
- Retrieving File Metadata
- Built-in File Management Functions
- Using print and write
- File Test Operators
- Directory Manipulation Using opendir, closedir, readdir, chdir, mkdir and rmdir

Implementing Command Line Arguments

- Reading Command Line Arguments from @ARGV
- Read Files Explicitly with <ARGV> and Implicitly with <>
- Manipulating Positional Parameters with push, pop, shift
- Process Lists of Files
- Processing Command Line Options with getopt or getopts
- Analyzing Command Line Argument Values with the Getopt::Std and Getopt::Long Modules
- Reserved Variables
- Manipulating Identifiable Options Using GetOptions

Debugging In Perl

- Using the Built-in Perl Debugger
- Starting the Debugger
- Debugger Command Syntax
- Checking for Script Syntax Errors
- Solving Compile-Time Errors
- Single-Stepping through a Script
- Executing to Breakpoints
- Setting Global Watches
- Printing Values of Variables
- Listing All Variables Used in the Script
- Using Strict Error Checking
- Quitting the Debugger

Input/Output Processing

- Parsing Input
- Using Standard Input, Standard Output, and Standard Error
- String and Field Processing
- Using Streams and Pipes
- Using die to Quit with an Error
- Redirecting Standard Output and Standard Error to a File
- Getting Standard Input from a File

Perl Report Formatting

- Defining Report Formats
- Justifying Text (Left, Right, Center)
- Using write to Generate Reports
- Defining here Documents for Report Customization
- Creating Report Headers
- Using Built-in Variables to Control Report Appearance
- Printing Line Numbers on a Report
- Formatting Multi-Line Output
- Writing Formatted Text to a File

References

- Life Cycle of a Reference
- Hard References and Anonymous References
- Use of References to Create Complex Data Structures
- Creating Hard and Anonymous References
- Modifying References
- Dereferencing a Reference
- The Arrow Operator ->
- Building Complex Data Structures with Multi-Dimensional Arrays & Hashes

Accessing a Database Using Perl DBI

- Database Access Life Cycle
- Using DBI and DBD to Connect to a Database
- Fundamental Data Storage and Retrieval Strategies
- DBI Query Syntax
- Using DBI Methods to Retrieve Database Information
    - Preparing Queries to be Executed
    - Creating Parameterized Queries
    - Executing Queries Using execute and do
- Fetching the Result Set to Achieve Workable Data in the Perl Script
    - Extracting Data Using an Array
    - Extracting Data Using a Hash
- Useful Utilities to Aid in Database Development
- Using Other Modules to Access Databases on the Web
- Extracting Data Using a Hash
- Displaying Results from Queries in a Report
- Releasing Database Resources

Perl Object Oriented Programming

- Object Oriented Programming Concepts
- Object Oriented Programming Terminology
- How Perl Implements Object Oriented Programming
- Modeling Software Objects Using Classes and Base Classes
- Creating Classes, Objects, Methods and Attributes
- Writing Constructors to Initialize of Objects
- Using bless to Turn References into Objects
- Creating Class Hierarchies through Inheritance

Web Architecture and CGI Scripting Overview

- Static vs. Dynamic Web Pages
    - Serving a Static HTML Web Page
    - Serving a Dynamic HTML Web Page
    - Dynamic Web Page Capabilities
- The Common Gateway Interface
    - How Server-Side CGI Scripts Work
- Differences between Client-Side and Server-Side Script Environment
- Strengths and Weaknesses of Web Programming Languages

CGI Scripting with Perl

- Perl's Role in Distributed Web Applications
- Using Environment Variables to Control CGI Scripts
- Communicating with the Web Server
- Perl CGI Script Instantiation and Invocation
- Generating Output for the Browser
- CGI Security Mechanisms

Configuring a Web Server for CGI

- Servers and CGI

Fundamentals of CGI Scripting with Perl

- Perl's Role in Distributed Web

- The Apache httpd.conf File
- Aliasing Standard Directories in Apache
- Using Standard Apache-Aliased Directories
- The Default Apache cgi-bin Directory
- Aliasing CGI-Enabled Directories in Apache

Applications

- Using Environment Variables to Control CGI Scripts
- Communicating with the Web Server
- Perl CGI Script Instantiation and Invocation
- Generating Output for the Browser
- CGI Security Mechanisms
- The Importance of the "Shebang" Line
- "Shebang" Line CGI Errors
- Debugging CGI Errors
- Linux vs. Windows File Format Errors
- Setting Linux File Protections
- Specifying the Page MIME Type
- Generating Standards-Compliant HTML

Generating Dynamic Web Pages Using Perl/CGI

- Generating Dynamic Web Pages and Dynamic Content
- The Role of JavaScript
- Generating JavaScript Using CGI
- Displaying Data in Tables
  - Using Environment Variables to Control CGI Scripts
- Displaying Data from Files
- CGI Output Stream Buffering

Dynamic Behavior Based on Query Strings

- The Query String Part of a URL
- Parsing the Query String
- Using Query Strings to Maintain Session State
- User-Defined Modules and CGI

Processing HTML Forms with Perl CGI

- Creating and Submitting HTML Forms
- HTML Input Elements
- Submitting a Form
- Form Interaction Summary
- Populating Form Elements Using CGI
- Processing HTML Forms
  - Characteristics of the GET Method
  - GET Method Environment Variables
  - Characteristics of the POST Method
  - POST Method Environment Variables
- Using Form Data Validation to Check Data

Using the Perl CGI.pm Module - Introduction

- Orientation to CGI.pm
- The Power of Perl-Supplied Routines
- Simplifying Debugging
- Using CGI.pm in Object-Oriented Style
- Named Parameter Syntax
- Using CGI.pm in Procedural Style
- Importing Groups of CGI.pm Methods
- CGI.pm Output and XHTML

Values

Form Processing with CGI.pm

- Classes of CGI.pm Methods
- Specifying Arguments for HTML Tags
- Differentiating Blank & Missing Parameters
- Using Passed Parameters as Numbers

Performance Optimization Using ModPerl

- Overview of Apache Web Server Functionality
- Comparing the Speed of CGI Scripting vs. ModPerl
- Configuring Apache with Perl and ModPerl
- Apache Strengths and Limitations
- Using the Apache::Registry Module
- Extending and Enhancing Apache Functionality

- Mixing CGI.pm Methods with Standard Perl
- Custom Tag Generation

Maintaining State with CGI.pm

- State Information and the Web
- Using Extra Path Information
- Using Hidden Fields in Forms
- Using Client-Side Cookies
- Setting Cookies
- Cookie Parameters
- Setting Cookies from Form Data
- Deleting Cookies
- Form Validation and Cookie Storage
- Cookie Deletion
- Overcoming Cookie Limitations

Developing Multi-Tiered Web Applications

- Review of Multi-Tiered Web Application Components
- Putting It All Together
- Using High-Level Packages to Assist with Scalability and Maintainability