Ruby Programming

Duration:	3 days
Type:	intermediate

Description

Ruby is a fully object oriented, dynamically typed scripting language. It has matured and expanded rapidly since its power was showcased by the 'Ruby on Rails' web framework. Today it is both the de facto replacement for the Perl scripting language and an increasingly important part of the Java and .NET platforms (via the JRuby and IronRuby interpreters).

This course provides experienced developers with everything they need to start using Ruby in production applications. The complete syntax of the language is covered, including detailed discussion of patterns, idioms and programming styles. Following this the course explores standard libraries and tools, and concludes with in-depth coverage of the JRuby interpreter and Java/Ruby integration.

Prerequisites

Delegates must be experienced software developers with a minimum of three years programming experience in languages such as Java, C#, Perl and Python.

List of Modules

Introduction to Ruby

- A brief history of the Ruby language
- Comparing Ruby to Perl and Python
- The available Ruby interpreters
- Scenarios where Ruby is best

Key Characteristics of Ruby

- Everything is an object
- Dynamically typed
- Flexible syntax and DSL's
- Lambdas and closures
- Metaprogramming support

Basic Ruby Constructs

- Types, literals and keywords
- Block structure and operators
- Conditionals and loop constructs
- Working with strings and characters
- The symbol table and symbol literals
- Declaring and running regular expressions
- The array and hash collection types

Basic Object Orientation

- Classes, objects and references
- Declaring classes with methods
- Adding fields via constructors
- Using attributes to define accessors
- Visibilities supported by Ruby
- Testing objects for equality
- Converting between object types
- Creating derived classes
- Overriding inherited methods

Advanced Object Orientation

- Using modules as namespaces and mixins
- Declaring class level fields and methods
- Creating class instance variables
- Understanding singleton methods
- Defining constant values in classes
- Operator overloading and aliases
- Using variable length argument lists
- Raising and catching exceptions

Procs, Lambdas and Closures

- Passing block arguments into methods
- Creating and invoking procs and lambdas
- Defining lambdas as literals in Ruby 1.9
- How closures extend the lifetime of variables
- Practical examples of using closures

Reflection and Metaprogramming in Ruby

- Examining an object via introspection
- Using *Method* objects to list and call methods
- Handling unknown methods with *method_missing*
- Dynamically adding methods via *class_eval* and *define_method*
- Tracing type loading and program execution

Libraries Supplied with Ruby

- Loading, reading and manipulating files
- Creating multithreaded applications
- Manipulating dates and times

Advanced Programming in Ruby

- Applying functional programming styles in Ruby
- Using Ruby to create Domain Specific Languages

Useful Ruby Tools

- Listing, installing and updating RubyGems via the *gem* command
- TDD usingTest::Unit, Mocha and RSpec
- Integration testing Web Applications using Watir
- Automating your build environment using Rake
- Building web applications via 'Ruby On Rails'

The JRuby Implementation of Ruby

- Why does the Java platform need Ruby?
- Writing Ruby programs using JRuby
- Invoking JRuby scripts from Java
- Using and extending Java classes and interfaces
- Converting between Ruby and Java types
- Using Ruby and JRuby in Ant buildfiles
- Building web applications via 'JRuby on Rails'