F Sharp Programming

F Sharp Programming

http://en.wikibooks.org/wiki/F_Sharp_Programming
This Book Is Generated By Wb2PDF
using
RenderX XEP, XML to PDF XSL-FO Formatter

08-04-2014

Table of Contents

F Sharp Programming

Contents[edit]

F# Basics[edit]

This section is suitable for complete beginners to F# and Functional Programming in general.

Working With Functions[edit]

F# is a functional programming language. Not surprisingly, functions are a big part of the language, and mastering them is the first step to becoming an effective F# developer.

Immutable Data Structures[edit]

"Data structure" is a fancy word which refers to anything that helps programmers group and represent related values in useful, logical units. F# has a number of built-in data structures which include tuples, records, lists, unions, and a number of others.

1. **Option Types** - Option types are simple, lightweight data structures which are commonly used to represent successful or failed computation.
2. **Tuples and Records** - Tuples and records are simple data structures which allow programmers to group related values together into a single unit.
3. **Lists** - A list represents an ordered group of values. F#'s List library has extensive support for manipulating and working with lists.
4. **Sequences** - Sequence expressions represent sequences of data computed on-demand.
5. **Sets and Maps** - Sets are conceptually similar to lists, except they cannot hold duplicate items. Maps allows programmers to relate *keys* to *values* and find items in the collection very efficiently.
6. **Discriminated Unions** - Discriminated unions represent a finite, well-defined set of choices. Discriminated unions are often the tool of choice building up more complicated data structures including linked lists and a wide range of trees.

Imperative Programming[edit]

F# is an "impure" programming language, meaning it allows programmers to write functions with side-effects and mutable state, very similar to the programming style used by imperative programming languages such as C# and Java.

1. **Mutable Data** - By default, variables in F# are immutable. However, F# supports mutable variables through mutable fields and ref cells.
2. **Control Flow** - Decision making and Loops.
3. **Arrays** - Arrays are ubiquitous mutable data structure used in imperative programming languages.
4. **Mutable Collections** - Lists and Dictionaries.
5. **Basic I/O** - Reading and writing to files and the console window.

6. **Exception Handling** - Exception handling allows programmers to catch and handle errors whenever an application enters an invalid state.

Object Oriented Programming[edit]

F# is a CLI/.NET programming language. CLI is an object-oriented platform. One of the most important features of F# is its ability to mix and match styles: since the .NET platform is Object Oriented, with F#, you often work with objects.

1. **Operator Overloading** - C#-like operator overloading.
2. **Classes** - classes and objects are the foundation of object-oriented programming (OOP). They are used to model actions, processes, and any conceptual entities in applications.
3. **Inheritance** - inheritance makes OOP code reusable. It allows programmers to create classes which inherit features from another class and add its own modifications.
4. **Interfaces** - interfaces abstract away the implementation details of a class by defining a template of methods an object must implement and expose publicly.
5. **Events** - events allow a classes to send and receive messages between one another.
6. **Modules and Namespaces** - modules and namespaces are used to organize classes into groups of related functionality.

F# Advanced[edit]

F# is easy enough for beginners to learn as their first language, yet it provides a powerful set of tools which can be appreciated by experienced developers. This section describes advanced syntactic contructs and techniques often used in F# programs.

1. **Units of Measure** - Units of measure attach metadata to floats, which allows floats to represent kilograms, pounds, Newtons, hectares, and so on.
2. **Caching** - Techniques to store computed values for efficient future retrieval. Also called Memoization.
3. **Active Patterns** - Active patterns allow programmers to wrap ad hoc values and objects in union-like structures for use in pattern matching.
4. **Advanced Data Structures** - Overview of techniques used to implement immutable data structures.
5. **Reflection** - Reflection allows programmers to inspect types and metadata in objects.

6. **Quotations** - Quotations convert arbitrary F# code into an abstract syntax tree.
7. **Computation Expressions** - Similar to monads in Haskell, computation expressions are used to simplify code written continuation-passing style.

Multi-threaded and Concurrent Applications[edit]

Multi-threading is becoming increasingly important with the development of multi-core processors. Functional programmers can take advantage of immutable data structures to make massively scalable, concurrent applications that are simple and easy to write.

1. **Async Workflows** - F#'s async primitive is fundamental for writing functional, simple multi-threaded code.
2. **MailboxProcessor Class** - Mailboxes are used to implement "message-passing concurrency," a style of concurrent programming used in massively parallel applications consisting of 10s or 1000s of independent nodes.

F# Tools[edit]

1. **Lexing and Parsing** - FsLex and FsYacc, lexer/parser generators based on the GNU Bison family of generators, are used to implement custom grammars and domain-specific languages in F#.

Authors[edit]

If you have contributed to this book, please add your name to this list.

1. Awesome Princess

Notes[edit]

Resources[edit]

- F# Language Reference on MSDN Library
- F# Homepage on Microsoft Research
- Microsoft F# Developer Center
- Real-World F# Articles on MSDN
- Language Specification
- Language Specification PDF
- F# Component Design Guidelines
- F# Component Design Guidelines PDF
- hubFS F# Community
- fpish community-driven events and learning material
- Community for F# monthly, virtual user group
- F# Snippets
- Try F# online
- Using the F# Language for Teaching
- Cross-platform and other F# extensions
- F# source code and community projects on GitHub
- F# cross-platform packages and samples
- The F# Survival Guide
- F# for fun and profit
- F-Sharp Wiki **(No longer available)**
- Objective Caml
- F# for game development
- Learning F# Through Game Development with XNA
- F# Tutorials for Beginning through Advanced Learners