Quick Comparison of C Sharp and Ruby

Adrian Kosmaczewski

2006-05-05

I have been working as a software developer since 1996, and as such I've used a variety of different languages, both compiled and interpreted. But the who languages that I know and use most today, are two somewhat different ones, C# and Ruby. I will begin my presentation with a short explanation of both, providing their major similarities and differences, and then providing some code samples of both.

Both languages are ranked #7 and #21 respectively in the TIOBE Programming Community Index, as of February 2006 (http://www.tiobe.com/tpci.htm).

The C# Programming Language

C# was created by Microsoft as part of its .NET programming environment, and while not oficially acknowledged, it is deeply inspired in Java:

> "C# (pronounced"C sharp") is a simple, modern, object-oriented, and type-safe programming language. It will immediately be familiar to C and C++ programmers. C# combines the high productivity of Rapid Application Development (RAD) languages and the raw power of C++."

(Microsoft, 2000)

It was designed by Anders Hejlsberg, who also created the well-known programming environments of Turbo Pascal and Delphi, while he worked for Borland before joining Microsoft. Even if being a propietary language, its core elements have been standardized by the ECMA and the ISO standard bodies, respectively as the ECMA-334 and ISO/IEC 23270 standards.

Applications written in the C# language run inside the CLR, which is the name of the virtual machine environment; there are several implementations of the CLR, one being .NET itself, the other being the open-source Mono runtime (http://www.mono-project.com/)

C# is a compiled, strongly-typed, object-oriented language. In C#, everything is an object, unlike Java where, for example integer variables and other basic types are not objects per se. Memory management is done automatically by a "garbage collector" facility built into the CLR, much like Java does. Actually, C# provides single inheritance with interface support, that is, exactly the same approach that Java provides. All these similarities, coupled with the syntax similarity, shows clearly the common design principles of both programming languages.

I have been using C# in my day-to-day job since 2002, and as such I've found the following features as the most useful:

- A particularly "picky" compiler: the C# compiler performs a high number of checkings which avoid common run-time problems related to type casting. In fact, C# does not allow all types of type casting to be done automatically, and the warnings provided by the compiler help the developer to create more stable and maintainable applications.
- Runtime capabilities: even being a language with a strong compiler and static typing, the .NET runtime is also capable of lots of operations by itself, mostly using reflection capabilities, and C# makes using these capabilities very easy. These are interesting but also dangerous, since they also can have a strong impact on performance.

The Ruby Programming Language

Ruby was created in 1993 by Yukihiro Matsumoto. It is an interpreted, dynamically typed, object-oriented language available in nearly all hardware and software platforms, for free; indeed, Ruby is an open-source programming language, for which its implementation can be freely used in any context, without restrictions of any kind.

Ruby has been inspired from Smalltalk, Perl and Python, and designed from the very beginning with the developer in mind:

Matz's primary design consideration is to make programmers happy by reducing the menial work they must do, following the principles of good user interface design. He stresses that systems design needs to emphasize human, rather than computer, needs: Often people, especially computer engineers, focus on the machines. They think, "By doing this, the machine will run faster. By doing this, the machine will run more effectively. By doing this, the machine will something something something." They are focusing on machines. But in fact we need to focus on humans, on how humans care about doing programming or operating the application of the machines. We are the masters. They are the slaves.

(Wikipedia, 2006)

Ruby also supports single inheritance, but instead of providing interfaces, it provides "mixins", which allow a class to use the functionality defined in modules without having to subclass them.

I started using Ruby last year (2005) because of the discovery of the overhyped application framework “Ruby on Rails” (http://www.rubyonrails.com/) that allows the rapid development and deployment of web-based applications using the Ruby language. I had the opportunity to use Ruby on Rails in a couple of applications and had an unparalleled level of productivity with it.

Similarities and Differences

Like C#, Ruby is a pure object-oriented language; in both languages, everything is an object. Both languages use exception handling to notify errors, allow object introspection, both have automatic memory management through garbage collection and allow the creation of multithreaded applications. They are also very “hyped” languages, which have a huge level of support in the industry, by different vendors and communities.

The code below does exactly the same task in both languages: (Download files)

In Ruby:

```ruby
# To execute, type "ruby employee.rb" at the command line
# (tested with Ruby 1.8.2)

class Employee
    def initialize(name, age)
        raise "Cannot be so young!" if age < 0
        raise "Cannot be so old!" unless age < 130
        @name, @age = name, age
    end

    def greet
        puts "Hi, my name is #{@name} and I am #{@age} years old"
    end
end

begin
    raise "Where are the parameters?" if ARGV.length < 2
    name, age = ARGV[0], ARGV[1].to_i
    employee = Employee.new(name, age)
    employee.greet
rescue
    puts "There was an error: " + $!
ensure
    puts "The program finished!"
end
```

In C#:

```csharp
/*
To compile this program, just type
```

```
"csc employee.cs" in your command prompt;
this will generate an "Employee.exe" in the same path
(tested with the .NET Framework 2.0)
*/

class Program
{
    static void Main(string[] args)
    {
        try
        {
            if (args.Length < 2)
            {
                throw new System.Exception("Where are the parameters?");
            }
            string name = args[0];
            int age = int.Parse(args[1]);
            Employee employee = new Employee(name, age);
            employee.Greet();
        }
        catch (System.Exception e)
        {
            System.Console.WriteLine("There was an error: {0}",
                                          e.Message);
        }
        finally
        {
            System.Console.WriteLine("The program finished!");
        }
    }
}

class Employee
{
    private string name = string.Empty;
    private int age = 0;

    public Employee(string name, int age)
    {
        if (age < 0)
        {
            throw new System.Exception("Cannot be so young!");
        }
        if (age &gt; 130)
        {
            throw new System.Exception("Cannot be so old!");
        }
        this.name = name;
        this.age = age;
    }

    public void Greet()
    {
        System.Console.WriteLine("Hi, my name is {0} and I am {1} years old",
            this.name, this.age);
    }
}
```

The most important difference among C# and Ruby is typing. Indeed, C# is a statically-typed language while Ruby is a dynamically-typed one. The dynamicity of Ruby has the advantage of making programs shorter and more readable, but has also the drawback of being a trapdoor, causing hard-to-find run-time errors, specially when used by inexperienced developers. In the above example, the C# program will raise an exception if you specify a string instead of a number for the age... while the Ruby interpreter will not complain!

On the other side, the Ruby code is roughly half the length of its C# counterpart; in general, you need less lines of Ruby code to do any task, and this is confirmed in the blog "Following the rewrite" (http://rewrite.rickbradley.com/), where there is a follow-up of a rewrite of a Java application in Ruby on Rails:

> Preliminary tests by our Technical Lead put the code reduction for a normal module in our Java stack converted to Rails at roughly 20:1.

(Rick Bradley, 2005)

And whoever says less lines of code, says also less lines to read, understand and maintain in the future.

Conclusion

Each language has its advantages and disadvantages; in the case of C# and Ruby, I find both languages to be perfectly complementary, while having some very important characteristics in common. I think that both languages will be more used in the future, and that the interoperability between them will be greater as well.

References

Microsoft, "The C# Language" [Internet], http://msdn.microsoft.com/vcsharp/programming/language/ (Accessed February 26th, 2006)

Rick Bradley, "Evaluation: moving from Java to Ruby on Rails for the CenterNet rewrite" [Internet], http://rewrite.rickbradley.com/pages/moving_to_rails/ (Accessed February 26th, 2006)

Ruby web site [Internet], http://www.ruby-lang.org/ (Accessed February 26th, 2006)

TIOBE Programming Community Index [Internet], http://www.tiobe.com/tpci.htm (Accessed February 26th, 2006)

Wikipedia, “C Sharp” [Internet], http://en.wikipedia.org/wiki/C_Sharp_programming_language (Accessed February 26th, 2006)

Wikipedia, “Ruby programming language” [Internet], http://en.wikipedia.org/wiki/Ruby_programming_langu
(Accessed February 26th, 2006)