Learning C# – the “top ten” list

When we begin to learn a new computer programming language, we often try to get acquainted with the language by applying our current knowledge to its characteristics. This “top ten” list captures this process.

As a second-year Software Development student, you have some experience with two or three programming languages. Let’s begin to learn about C# by asking questions in the form of a “top ten” list. (Note that you can ask these questions about **ANY** language.)

Note: C# is the name of the programming language.

The .NET Framework is the name of the class library and execution runtime. The classes are organized into namespaces. Your app will link to one or more namespaces.

So in summary, you write code in the C# language, and use .NET Framework classes.

The “top ten” list of questions

Here is the “top ten” list of questions to ask when learning C#:

1. How do I create/edit/save the source code? What editor do I use? Where do I store the source code? What is the source code file naming convention?

2. How do I build/compile and then execute/run my program? Is there a compiler? Is there a host execution (aka “runtime”) environment?

3. What is the program’s entry point? Syntactically, what coding convention must I follow?

4. What data types can I use? Are they categorized (e.g. value/stack, reference/heap), and if so, what do I need to know about their characteristics (e.g. size, initial value, precision, convertibility, etc.)?

5. How do I declare new variables/fields of a data type? How do I instantiate them? How do I refer to and use these variables/fields? (Note that a type could be a class declaration for an object-oriented language and platform.)

6. What operators are available to me in this language? (How do they compare to other languages that I’m familiar with?)

7. What is the syntax of a statement, and of an expression? What delimiters are used? How is scope and hierarchy expressed?

8. What syntax enables me to process a decision (e.g. if-else, or switch-case)?

9. What syntax enables me to process repeatedly (e.g. for, while)?

10. How are functions (methods) declared? What rules are there for argument/parameter declaration? How are functions used/called? Are the arguments passed by value, or by reference?

Bonus question:

11. Where do I get learning/tutorial information? Where do I get reference information? Where do I look for good-quality online information?

.

The answers

Here are the answers. We cover them in class, and we go through some examples.

1. How do I create/edit/save the source code? What editor do I use? Where do I store the source code? What is the source code file naming convention?

For C# ASP.NET MVC applications…

- Visual Studio is used
- You work with a local project, and deploy it when ready
- Code classes are found in the Controllers and Models folder, and elsewhere
- Code classes have a “cs” filename extension
- Source code for user interface “views” is found in the Views folder, in files that have a “cshtml” filename extension

2. How do I build/compile and then execute/run my program? Is there a compiler? Is there a host execution (aka “runtime”) environment? Does it control the application lifecycle?

In Visual Studio, there are several ways:

1. Build, then File > View in browser

2. Ctrl+F5 (or simply F5 to start the debug environment)
3. In Solution Explorer, right-click the project, then View in browser

Your app is compiled. It runs on the ASP.NET runtime environment, which provides services for web apps. In turn, ASP.NET depends upon the .NET Framework's "common language runtime".

By default, when your app loads for the first time on a host, it has a 20-minute timeout. Every new request to a resource in the app will reset the timer.

3. What is the program's entry point? Syntactically, what coding convention must I follow?

The ASP.NET runtime has an entry point. Ultimately, it does reduce down to a "main" method in a "main.c" module on the host.

Closer to your app, the app's execution domain is based on its URL. An incoming request is examined by the "routing engine".

If the request is for a valid and returnable resource (e.g. an image, or a CSS source code file), then the resource is returned.

However, if not, then the routing engine will locate the controller and method that will service the request. The controller class is instantiated, and then the appropriate method executes.

4. What data types can I use? Are they categorized (e.g. value/stack, reference/heap), and if so, what do I need to know about their characteristics (e.g. size, initial value, precision, convertibility, etc.)?

The .NET Framework offers its languages a "common type system". They include a selection of "value" types (where the type's data is a fixed size, and lives on a stack), and "reference" types (where the type's data varies in size, and the stack content is a pointer to the type's data in "heap" memory).

All .NET Framework types are objects, even types you think of as scalars in other languages. It is a rich type system, and we'll learn about and use the types as we go along.

We also create our own types (classes).

.

5. How do I declare new variables/fields of a data type? How do I instantiate them? How do I refer to and use these variables/fields? (Note that a type could be a class declaration for an object-oriented language and platform.)

Examples…

Declaration syntax:

typeName variableName;

Instantiation syntax:

typeName variableName = new typeName(constructor, arguments, go, here);

Referring to and using syntax:

variableName = someValue;

Response.Write(variableName.ToString());

6. What operators are available to me in this language? (How do they compare to other languages that I'm familiar with?)

A rich set of operators are available, using a comfortable syntax for C programmers.

7. What is the syntax of a statement, and of an expression? What delimiters are used? How is scope and hierarchy expressed?

C# is a C-family language, so its statement syntax is similar.

Expressions are also similar, and you can use operators on objects without conversions.

Statements end with a ; semicolon, just like C.

Curly braces { } are used as code block delimiters.

Scoping follows rules similar to C. For type and member (e.g. methods, properties) visibility, there is a default, but there are scoping keywords that can be used to modify the scope.

.

8. What syntax enables me to process a decision (e.g. if-else, or switch-case)?

C# is a C-family language, so its statement syntax is similar. if-else (including the ? : syntax) and switch-case are available.

.

9. What syntax enables me to process repeatedly (e.g. for, while)?

C# is a C-family language, so its statement syntax is similar. for and while are available. There is also a foreach enumerator form of the for statement.

10. How are functions (methods) declared? What rules are there for argument/parameter declaration? How are functions used/called? Are the arguments passed by value, or by reference?

In a class, methods are declared like this:

scope returnTypeName methodName(typeName argument1, typeName argument2) { }

Arguments must include a type name declaration. There are options for parameters for directionality, defaults, and by-reference (mentioned next).

In C#, pass-by-value is the default. The method receives a copy of the value from the caller. If you wish, you can change this, per argument.