Next-Generation Prog. Langs.

8-26-2010

Opening Discussion

- Welcome to class.
- How was your summer?
- Have you had any experience with Scala/F#/X10/Fortress or anything else that you might consider next-generation?

Schedule

- We basically run through the Scala book then the F# book.
- After that there will be student presentations on other languages.

Syllabus

- Let's go over the syllabus.
- The beginning part is fairly straight forward.

Course Format

- This course has a different format than you might be used to.
- I'm not preparing lectures after today. Instead I will moderate discussion.
- The discussion will be based on things you bring to class based on the reading.

Grades

- Your grade comes from the combination of four different areas.
  - Projects (2) - 40%
  - Daily Code – 25%
  - Daily Questions – 20%
  - Presentation – 15%

Projects

- You will turn in two projects.
- They can be done individually or in groups.
- I have provided a few ideas. The projects are very open.
- They just need to be sufficient scope for the number of people working on them.
- Each person will turn in a paper for each project.

Daily Code

- Every class day people will bring code to class. It can be whatever you want, but it needs to involve the topic from the day.
- The class will be split in two for this and they will alternate.
- The first half of the discussion is based on the code you bring in. Inventive code will make better discussion.
- This code can be something useful for the project.

Daily Questions

- All students will also bring in two questions for each class over the reading material.
- These will be used as the seeds for the rest of the discussion.
- The quality of your questions factors into your grade.

A Scalable Language

- The name Scala stands for Scalable Language.
- The language itself is not extremely large.
- It provides the capability to add libraries that appear to be language extensions.
  - Pass-by-name
  - Methods as operators
- Scala also supports programming in the small as well as in the large.
  - REPL
  - Scripting

Fully Object-Oriented

- All values in Scala are objects (primitives will be non-objects as optimization, but it doesn't impact programming).
- There is no static. Instead you use companion objects.
- All operators are method invocations using operator syntax.
  - 1+2 is really 1.+(2)

Highly Functional

- Scala includes many functional aspects.
- Functions are first-class values. They can be passed around freely.
- Almost everything is an expression.
- You have function literals and higher-order functions.
- You can curry functions or do partial applications of functions.
- Lazy evaluation.
- Pass-by-name.

Java Compatibility

- Scala compiles to Java bytecode and can run on any system with Java.
- Allows seamless calling of Java code.
- Java code can generally call Scala code as well.

Concise and Expressive

- Scala code is typically very concise. You can express a lot without too many keystrokes.
- It is highly expressive so you can say a lot with a little.
- This is done in a generally readable manner. Too concise/expressive tends to lead to obfuscation.

High-Level

- You express ideas at a high level.
- Spend more time saying what you want, not how you want it done.
- This has a tendency to reduce bugs and can make the code easier for other people to read and understand.

Static Typing with Type Inference

- Like Java, Scala is type safe with most of the type checking done statically.
- Unlike Java, Scala has local type inference. As a result, you very rarely specify types in Scala. Most of the time you let it figure it out.
- This makes code cleaner and easier to write.

The Scala Interpreter

- The command scala can be used to bring up an interpreter if no file is specified.
- This puts you in a REPL that can be used to quickly test how things work.
- This is the ultimate form of programming in the small.
- You can also load in files to test things.

Variables

- There are two keywords for declaring variables in Scala.
  - val – This is like a final variable in Java. By default you should use this.
  - var – This is like a normal variable in Java. It can be changed.
- Whereas in most languages you are used to the type comes before the name, in Scala it follows it. If it is needed it comes after a colon.
- You must initialize variables at declaration.

Functions

- The keyword def is used to declare functions.
- Arguments go in parentheses like normal. Types go after names separated by a colon.
- The arguments of a function require types.
- Return type recommended though can be inferred.
- Equals sign unless returns Unit.
  - def func(a1:T1,a2:T2):RType = expr.
- Return value of last expression.

Scripts

- You can write simple programs in files that end in .scala and then execute them with the scala command followed by the file name.
- No need for a main.
- Lines are executed in order from the top.
- This is part of the programming in the small support you get in Scala.
- Unlike Java it works well for small programs.

Familiar Constructs

- If looks like you are used to, but is an expression. So no ternary operator is needed.
- The while loop and do-while loop are just like you are used to. They are the only statements in Scala that aren't expressions.

Different For Loop

- The for loop is a bit different in Scala.
- It is always a for-each loop that goes through a collection.
    - for(e ← coll)
- You can easily use it for counting by using a Range object.
    - for(i ← 1 to 10)
    - for(i ← 0 until 10)
- Yield makes it an expression.

Playing Around

- With any time that might be left we can pull up Scala and play with it.