Theory: Overview of the basic program

6 minutes 0 / 3 problems solved

We've been only formally introduced to Scala, so now it's time to say "Hello". In this topic, we will look at a particular Scala program and see what's going on there. We will touch on aspects related to the term **program** in general and some basic code components. Even though our program will be somewhat basic, it still has to work correctly and effectively.

§1. Hello, Scala!

Take a look at the source code of our program:

```
object HelloScala extends App {
   println("Hello, Scala!")
}
```

This program simply prints "Hello, Scala!". Note that typically, the source code is placed in a file, transformed by the compiler and only then you can run the program. With the help of web-compilation you can run it here, in the browser.

§2. REPL

Sometimes you want to experiment with the code of your programming language: check a tricky expression, prove some hypothesis, or calculate something. Typically, such actions are done by placing the code inside the test files. However, there is another option: **read–eval–print loop (REPL)**, a shell where you can interactively input your code line by line and receive results, for example:

```
scala> 1 + 2
res0: Int = 3
```

You can easily check the printing part of our first program and get the result:

```
scala> println("Hello, Scala!")
Hello, Scala!
```

REPL helps to avoid the routine of file management, which is convenient. That's why we introduce this feature: we use it a lot with Scala language code examples.

§3. Program interpretation

Let's take a step back and try to look at our program in general. There are different ways to do that.

First, a program is a set of instructions executed sequentially: instruction1, instruction2, ... and each instruction changes the state of the program. This view called **imperative programming** style.

```
print("Hello, Scala!")
println()
```

We could also think about a program as a process of simplifying a flow of instructions, like instruction1 + instruction2 => instruction3 (we can call it **reduction**).

```
print("Hello, Scala!" + "\n")
```

This kind of view relies on **functional programming** in general and on Scala in particular.

Current topic:

Overview of the basic program

Topic depends on:

× Introduction to Scala

Topic is required for:

Basic literals

Table of contents:

§4. A closer look at Hello, Scala

Let's look more closely at the specific details of our first program.

The entry point. We should know where our program starts to refer it by some code identity (in our first program, it is HelloScala). For Scala, it is some code that extends the App (application). If you're familiar with Java and the concept of main function, then you'll see that it's basically the same idea:

```
object Main {
  def main(args: Array[String]): Unit = {
    println("Hello, Scala!")
  }
}
```

App is one of the advantages that Scala has over Java: it makes it possible to write program code inside the body of an entry point.

Data transformation. Our program is a set of instructions that transforms command line arguments to program exit code. In Scala, we usually define this inside the App code block. As mentioned before, you could interpret it differently: execution of a set of instructions step-by-step or simplification of instructions.

Endpoint. Every program terminates at some point (unless we're dealing with an infinite loop of execution). Our first program is not an exception: it finishes right after printing the message.

§5. Conclusion

Congratulations!

We have written our first program, which prints "Hello, Scala!", learned about the REPL code shell, and considered programs as executing a set of instructions step-by-step or simplifying the instructions. We have also described the core points of any Scala program: entry point, transformations and endpoint.

Feel free to use our first code experience as a template for more complex tasks.

Report a typo

26 users liked this theory. **3** didn't like it. **What about you?**

Start practicing

This content was created about 1 year ago and updated 3 days ago. Share your feedback below in comments to help us improve it!

Comments (2) Hints (0) Useful links (0) Show discussion