Organize Code

Effective Programming in Scala

Worksheets vs Projects

Scala worksheets are defined in a single file, but as a program grows it can be more convenient to implement it as a Scala "project" containing multiple files.

In this lesson and the following ones, we will show how to organize Scala projects: where do we write the source files? How do we refer to definitions written in other files? etc.

Names

We have seen that definitions introduce names that can be reused later to refer to the content of the definitions.

However, introducing names comes with drawbacks:

- **Collisions** (or clashes), in case we want to give two different things the same name,
- **Coupling** with the parts of code that refer to the names.

Packages

Packages allow you to give a prefix to your definitions so that they don't collide.

```
// File areas.scala
package area

val facade = 5 * 3

// File prices.scala
package price

val paint  = 3.5
val facade = area.facade * paint
```

Packages Clauses

To place a definition inside a package, use a package clause at the top of your source file.

```
package effective.example

object Hello:
  val foo = ...
```

This would place the object `Hello` in the package `effective.example`.

You can then refer to `Hello` by its *fully qualified name* `effective.example.Hello`.

Packages Tree

It is a good practice to put the source files in subdirectories mirroring the packages structure.

For instance, the object `effective.example.Hello` should be defined in a file located at path `effective/example/Hello.scala`.

Note: since we usually define source directories in a folder `src/main/scala`, the full path of the file `Hello.scala` (relative to the project root directory) would be:

```
src/main/scala/effective/example/Hello.scala
```

Imports

Referring to names by fully qualifying their package prefix can be cumbersome.

You can **import** names as follows:

```
import effective.example.Hello

println(Hello.foo)
```

Forms of Imports

Imports come in several forms:

```
import effective.example.Hello          // imports just Hello

import effective.example.Hello.foo      // imports just foo

import effective.example.{Hello, Bar}   // imports both Hello and Bar

import effective.example.*              // imports everything from
                                        // package effective.example
```

The first three forms are called *named imports*.

The last form is called a *wildcard import*.

You can import from either a package or an object.

Wildcard Import (Scala 2 Compatibility)

In Scala 2, the wildcard import used the character “underscore” instead of “star”:

```
import effective.example._
```

Standard Library

The definitions introduced by the standard library live in the `scala` package.

For instance, `scala.util.Random`, `scala.Int`, or `scala.collection.immutable.List`.

Predefined Imports

Some entities are automatically imported in any Scala program:

- All members of package `scala`
- All members of package `java.lang`
- All members of the singleton object `scala.Predef`.

This is why you can just write `Int`, for instance, instead of `scala.Int`.

Scaladoc

You can explore the standard Scala library using the scaladoc web pages.

Start at www.scala-lang.org/api/current

Summary

Large Scala projects are split into multiple files.

Source files contain definitions organized into **packages**.

You can **import** entities to not have to write their fully qualified name every time.