

What is the current latest version of Scala? What is the major change or update in Scala 2.12?

Current Scala's stable is 2.11.7. It supports Java SE 7.

The major change or update in Scala 2.12 version is that it supports Java SE 8 or later versions only. Scala 2.12 is not a binary compatible with the 2.11.x series. It's still in Mile Stone Builds only.

What is Option in Scala? What are Some and None? What is Option/Some/None Design Pattern in Scala?

In Scala, Option is used to represent optional values that is either exist or not exist.

Option is an abstract class. Option has two subclasses: Some and None. All three (Option, Some and None) are defined in "scala" package like "scala.Option".

Option is a bounded collection in Scala, which contains either zero or one element. If Option contains zero elements that is None. If Option contains one element, that is Some.

Some is used to represent existing value. None is used to represent non-existent value.

Example:-

```
def get(val index: Int): Option[String]
```

Let us assume that this method is from List. This method has a return type of Option[String]. If List contains elements, this get method returns "Some[String]" element available in that index position. Otherwise, it returns "None" (that is no elements)

Some is a case class and None is an Object. As both are case class/object, we can use them in Pattern Matching very well.

The combination of all these three definitions is known as Option/Some/None Design Pattern in Scala.

What is Either in Scala? What are Left and Right in Scala? Explain Either/Left/Right Design Pattern in Scala?

In Scala, Either is an abstract class. It is used to represent one value of two possible types. It takes two type parameters: Either[A,B].

It exactly have two subtypes: Left and Right. If Either[A,B] represents an instance A that means it is Left. If it represents an instance B that means it is Right.

This is known as Either/Left/Right Design Pattern in Scala.

What is the equivalent construct of Scala's Option in Java SE 8? What is the use of Option in Scala?

Scala's Option is similar to Java SE 8's Optional. Java SE 8 has introduced a new utility class Optional to represent existing or non-existing of some value. Optional is available in java.util package.

Both Scala's Option and Java SE 8's Optional are used to represent optional values. Both are used to avoid unwanted null checks and NullPointerException.

What are the Advantages of Functional Programming (FP) or Advantages of Pure Functions?

The following are the Advantages of Functional Programming (FP) or Advantages of Pure Functions:

More Modular

Easier to understand Or Easier reason about

Easier to test

Less prone to bugs

Easier to reuse

Easier to Parallelism and generalize

What are the Popular Scala-Based Frameworks to develop RESTful Web Services or REST API?

There are many Scala-Based Framework to develop RESTful Web Services. Most popular frameworks are:

Play Framework

In Play, we call REST API URLs as routes. We place all routes at once place in Play framework. It is a stateless web framework to develop REST API easily.

Scalatra Framework

It is very simple and easy Scala-based web framework to develop REST API

Spray Framework

It is very concise and built on top of Akka framework so it's better to develop REST API using Actor Model.

Lift Framework

It allows routing using Pattern Matching concept.

What is the best Framework to generate REST API documentation for Scala-based applications?

Swagger is the best tool for this purpose. It is very simple and open-source tool for generating REST APIs documentation with JSON for Scala-based applications.

If we use Play with Scala to develop your REST API, then use play-swagger module for REST API documentation.

If we use Spray with Scala to develop your REST API, then use spray-swagger module for REST API documentation.

Like Hibernate for Java-based applications, What are the Popular ORM Frameworks available to use in Play/Scala based applications?

Like JPA, Hibernate and Toplink etc ORM Frameworks for Java-based applications, There are many ORM frameworks to use in Play/Scala based applications.

Popular ORM frameworks for Play/Scala based applications:

Slick

Anorm

SORM(Scala ORM)

Squeryl

What is the best tool to develop Play/Scala applications to persist data in MongoDB NoSQL data store?

ReactiveMongo is the best Scala Driver to develop Play/Scala applications to persist data in MongoDB NoSQL data store. It supports fully non-blocking and asynchronous I/O operations.

Popular clients who are using Play and Scala to develop their applications?

Thousands of clients are using Play and Scala in Production. The following list is the more popular clients who are using Play and Scala actively.

LinkedIn

The Guardian

Ocado

LuchidChart

GOV.UK

What is the best language to use with Play framework: Scala or Java?

Play 2 is completely written in Scala. If we use Java with Play framework, we need to face many issues because Java does not support full FP features.

Scala is the best option to use with Play framework to develop Highly Scalable, Better Performance with Concurrency/Parallelism and Low latency applications, because:

Play 2 is completely written in Scala.

It supports full FP features.

It is more expression language than Java.

It supports Akka Actor model very easily

It supports some new OOP feature like Traits.

Play's built-in templates are developed in Scala

How Scala supports both Highly Scalable and Highly Performance applications?

As Scala supports Multi-Paradigm Programming(Both OOP and FP) and uses Actor Concurrency Model, we can develop very highly Scalable and high-performance applications very easily.

What are the available Build Tools to develop Play and Scala based Applications?

The following three are most popular available Build Tools to develop Play and Scala Applications:

SBT

Maven

Gradle

What is SBT? What is the best Build Tool to develop Play and Scala Applications?

SBT stands for Scala Build Tool. Its a Simple Build Tool to develop Scala-based applications.

Most of the people uses SBT Build tool for Play and Scala Applications. For example, IntelliJ IDEA Scala Plugin by default uses SBT as Build tool for this purpose.

What are the available Unit Testing, Functional Testing and/or BDD Frameworks for Play and Scala Based applications?

The following are most popular available Unit Testing, Functional Testing and/or BDD Frameworks for Play/Scala Based applications:

Spec2

ScalaTest

ScalaCheck

Mokito

What is the best Code-coverage tool available for Play and Scala based applications?

SCoverage is the Code-coverage tool for Play and Scala based applications.

SCoverage stands for Scala Code-coverage tool. It has three separate plug-ins to supports the following build tools:

SBT

Maven

Gradle

What is the best Scala style checker tool available for Play and Scala based applications?

Like Checkstyle for Java-Based Applications, Scalastyle is best Scala style checker tool available for Play and Scala based applications.

Scalastyle observes our Scala source code and indicates potential problems with it. It has three separate plug-ins to supports the following build tools:

SBT

Maven

Gradle

It has two separate plug-ins to supports the following two IDEs:

IntelliJ IDEA

Eclipse IDE

Which IDEs support Play and Scala-Based Applications Development and how?

The following two popular IDEs support Play and Scala-Based Applications Development:

IntelliJ IDEA

Eclipse IDE

They support by using Scala Plugins like Eclipse IDE has a Scala IDE for Eclipse to support Play and Scala-Based Applications Development.

IntelliJ IDEA has a plug-in like “Scala Plugin for IntelliJ IDEA” to support “Scala, SBT and Play 2 Framework” based applications.

What is the default Unit and Functional Testing Framework for Play? What is the default Build Tool for Play? What is the Default Template Engine for Play? What is the built-in Web Server available in Play Framework?

Play Framework’s default Unit and Functional Testing Framework is Spec2. It is very easy to test Play/Scala based applications using Spec2 Framework.

Play Framework’s Default built-in template is “Twirl”. It was developed in Scala. By using these templates, we can develop Play/Scala based applications very easily.

The Built-in or Default Web Server available for Play Framework is Netty Server.

Why Scala is better than Java? What are the advantages of Scala over Java (Java 8)? Compare to Java What are the major advantages or benefits of Scala?

Because Scala supports the following extra features, it is better than Java 8:

Full FP Features

More Expression Language

Pattern Matching

Better support for Akka Actor Model

Automatic resolution for Inheritance Diamond Problem with Traits

Asynchronous and Non-blocking IO programming using Akka Framework

Fully Reactive Streaming API

What is an Anonymous Function In Scala? What is a Function Literal in Scala? What are the advantages of a Anonymous Function/Function Literal in Scala?

Anonymous Function is also a Function but it does not have any function name. It is also known as a Function Literal.

The advantages of a Anonymous Function/Function Literal in Scala:

We can assign a Function Literal to variable

We can pass a Function Literal to another function/method

We can return a Function Literal as another function/method result/return value.

What is an Higher-Order Function (HOF)?

Higher Order Function (HOF) is also a function but which performs one, two or both of the following things:

Take other functions as arguments

Return functions as their results

What are the differences between Case class and Normal Class?

Case class is also a class, however when we compare it with normal class, it gives the following extra features or benefits:

By default, Case-class constructor parameters are 'val'. We don't need to declare parameters with 'val'.

By default, Case-class constructor parameters become class fields.

These methods are added automatically: toString, equals, hashCode, copy, apply and unapply.

It automatically gets Companion object.

No need to use 'new' keyword to create instance of Case Class.

Easy to use in Pattern Matching.

All these features are added by Scala Compiler at compile-time. It is not possible with normal class.

What are the advantages of Play/Scala stack to develop web applications?

The following are the major advantages of Play/Scala stack to develop web applications:

Open Source

Play is an Open-source free-software framework to develop web applications.

Better Productivity

Play framework's Auto-reload feature improves Developer Productivity. No need to build, deploy and test our changes. Just do our changes and refresh the page to see our changes.

Stateless and Easy to Develop REST API

Play is HTTP based stateless model to serve web requests so it is very easy to develop REST API or RESTful Web Services.

Better Error-Handling

If we develop our web application using Play framework, it informs all errors in the browser in very useful format. It shows error message, the file location, line number where error occurred, highlighting the code-snippet to understand the error very easily.

High Performance and Better Scalability With Reactive

Play framework is developed by following Reactive design patterns and it is built on top of Netty server to utilize Non-blocking IO Feature. Because of this feature, we can develop very highly Scalable and performance applications very easily.

Easy to Extend

Play is very flexible framework and supports developing plug-ins very easy to extend its features and functionality.

Highly Concurrency and Better Parallelism

As both Scala and Play supports Functional Programming, it is very easy to develop Highly Concurrency and Better Parallelism applications very easily because FP supports Immutability, Pure Functions (Functions without side-effects), Pattern Matching, Actor Model etc.

Better Reusability, Easy to Test and More Modular

As both Scala and Play supports Functional Programming, we can develop more modular and reusable applications. It is also very easy to test more modular applications.

What are the Java's OOP constructs not supported by Scala? What are the Scala's OOP constructs not supported by Java? What are the new OOPs constructs introduced by Scala, but not supported by Java?

Java's OOP constructs, which are not supported by Scala:

There is no interface concept in Scala

There is no Enum concept in Scala

Scala's OOP constructs, which are not supported by Java:

OR

The new OOPs constructs introduced by Scala, but not supported by Java:

Scala Traits

Solving Inheritance Diamond Problem automatically.

What is call-by-name? Does Scala and Java support call-by-name? What is the difference between call-by-value and call-by-name function parameters?

Call-by-name means evaluates method/function parameters only when we need them or we access them. If we don't use them, then it does not evaluate them.

Scala supports both call-by-value and call-by-name function parameters. However, Java supports only call-by-value, but not call-by-name.

Difference between call-by-value and call-by-name:

The major difference between these two are described below:

In Call-by-name, the function parameters are evaluated only whenever they are needed but not when the function is called.

In Call-by-value, the function parameters are evaluated when the function is called.

In Call-by-value, the parameters are evaluated before executing function and they are evaluated only once irrespective of how many times we used them in that function.

In Call-by-name, the parameters are evaluated whenever we access them and they are evaluated each time we use them in that function.

Scala Syntax Differences

Call-by-value:

```
def myFunction(a: Int, b: Int) { }
```

Here both a and b are Call-by-value parameters to myFunction.

Call-by-name:

```
def myFunction(a: Int, b: => Int) { }
```

Here both a is a Call-by-value parameter and b is Call-by-name to myFunction.

What are the popular MVC frameworks for Scala Language to develop Web Applications?

The following are the most popular MVC frameworks available for Scala Language to develop Web Applications:

Play Framework

Scalatra Framework

Spray Framework

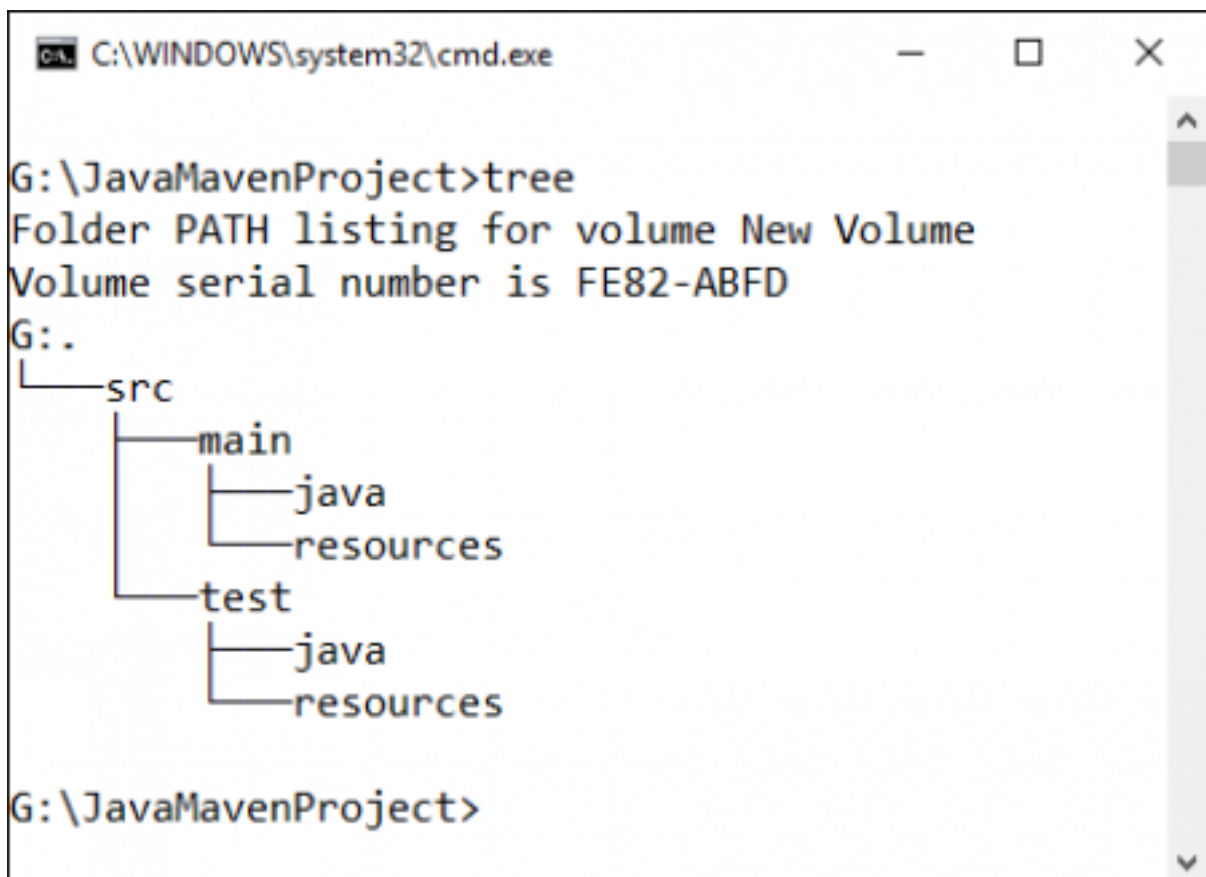
Lift Framework

What are major differences between Java-Based and Scala-Based Maven Project's structure?

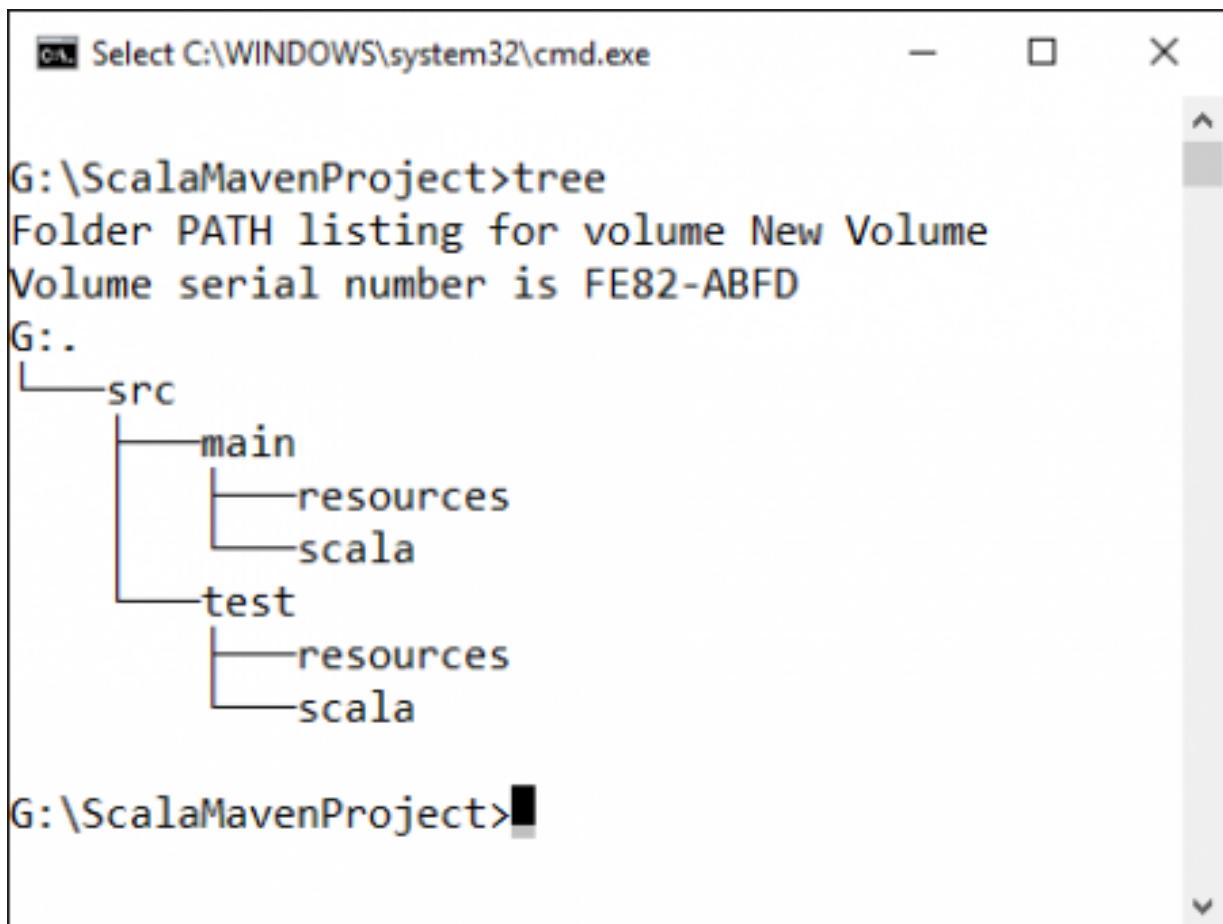
Most of the Java-based projects uses Maven as their Build tool. However, most of the people uses SBT as build tool to develop their Scala-based applications, but Some Teams uses Maven too as a build tool to develop their Scala-based applications.

Maven folder structure for both Java-based and Scala-based projects is almost same just one folder name change as shown in the below diagrams: java and scala folder names.

Java-based Maven Project Folder Structure:



Scala-based Maven Project Folder Structure:



```
Select C:\WINDOWS\system32\cmd.exe

G:\ScalaMavenProject>tree
Folder PATH listing for volume New Volume
Volume serial number is FE82-ABFD
G:
.
├── src
│   ├── main
│   │   ├── resources
│   │   └── scala
│   └── test
│       ├── resources
│       └── scala
G:\ScalaMavenProject>
```

What is Extractor in Scala? What is the difference between Constructor and Extractor in Scala? What is the use of Extractor in Scala?

Not only in Java and Scala, in almost all OOP languages Constructor is used to create (or assemble) an object or an instance of a Class using it's parameters (or components). Extractor is quite opposite to Constructor.

In Scala, Extractor is used to decompose or disassemble an object into it's parameters (or components).

In Scala, apply method is a Constructor. Internally, Extractor uses unapply method to decompose an objects into it's parts (or parameters). In Scala, Extractor is mainly used in Pattern Matching concept. We will discuss Pattern Matching concept soon.

What is the use of '???' in Scala-based Applications?

This '???' three question marks is not an operator, a method in Scala. It is used to mark a method which is 'In Progress' that means Developer should provide implementation for that one.

This method is define in scala.PreDef class as shown below:

```
def ??? : Nothing = throw new NotImplementedError
```

If we run that method without providing implementation, then it throws 'NotImplementedError' error as shown below:

```
scala> def add(a:Int, b:Int) : Int = ???
```

```
add: (a: Int, b: Int)Int
```

```
scala> add(10,20)
```

```
scala.NotImplementedError: an implementation is missing
```


Explain the main difference between List and Stream in Scala Collection API? How do we prove that difference? When do we choose Stream?

In Scala, both List and Stream are from Collection API and works almost similar. Both are Immutable collections.

However, there is one main difference between List and Stream in Scala Collection API: That is List elements are evaluated Eagerly and Stream elements are evaluated Lazily that means when we access them.

```
scala> var list1 = List(1,2,3,4)
```

```
list1: List[Int] = List(1, 2, 3, 4)
```

Here we can observe that all List elements evaluated at the time of creating List object. However, if we do same thing on Stream, we cannot see all elements. We can see only first evaluated element and remaining elements are evaluated lazily as shown below:

```
scala> var s1 = Stream(1,2,3,4)
```

```
s1: scala.collection.immutable.Stream[Int] = Stream(1, ?)
```

When we want Lazy collection to evaluate elements only when we access them then it's better to use Stream.

What is the difference between :: and #:: in Scala? What is the difference between ::: and #::: in Scala?

In Scala Collection API,

:: and ::: are methods available in List class.

#:: and #::: are methods available in Stream class

In List class, :: method is used to append an element to the beginning of the list.

```
scala> var list1 = List(1,2,3,4)
```

```
list1: List[Int] = List(1, 2, 3, 4)
```

```
scala> list1 = 0 :: list1
```

```
list1: List[Int] = List(0, 1, 2, 3, 4)
```

In List class, ::: method is used to concatenate the elements of a given list in front of this list.

```
scala> var list1 = List(3,4,5)
```

```
list1: List[Int] = List(3, 4, 5)
```

```
scala> val list2 = List(1,2) ::: list1
```

```
list2: List[Int] = List(1, 2, 0, 1, 2, 3, 4)
```

In Stream class, #:: method is used to append a given element at beginning of the stream. Only this newly added element is evaluated and followed by lazily evaluated stream elements.

```
scala> var s1 = Stream(1,2,3,4)
```

```
s1: scala.collection.immutable.Stream[Int] = Stream(1, ?)
```

```
scala> s1 = 0 #:: s1
```

```
s1: scala.collection.immutable.Stream[Int] = Stream(0, ?)
```

In Stream class, #::: method is used to concatenate a given stream at beginning of the stream. Only this newly added element is evaluated and followed by lazily evaluated stream elements.

```
scala> var s1 = Stream(1,2,3,4)
```

```
s1: scala.collection.immutable.Stream[Int] = Stream(1, ?)
```

```
scala> val s2 = Stream(-1,0) #::: s1
```

```
s2: scala.collection.immutable.Stream[Int] = Stream(-1, ?)
```

:: method works as a cons operator for List class and #:: method works as a cons operator for Stream class. Here 'cons' stands for construct.

::: method works as a concatenation operator for List class and #::: method works as a concatenation operator for Stream class.

If I want to become a Fullstack Scala Developer, which technology stack I should learn?

If you want to become a Fullstack Scala Developer, you should learn the following technology stack:

Scala 2.11.7

Play 2.4.6 Framework

Akka 2.3 Framework

One Build Tool: SBT/Maven

One JS Framework: CoffeeScript/JavaScript

One IDE: IntelliJ IDEA 15/ Eclipse IDE 4.x

One TDD & BDD Framework: ScalaTest,Spec2,ScalaCheck,Mockito

Micro Services with Play and Scala

SCoverage

Scalastyle

Functional Programming Design Patterns

Machine Learning with Scala

NOTE:- In Scala, Extractor follows Extractor Design Pattern. If you want to learn it in depth, please go through my Scala Tutorial (Most of the posts follow this pattern: Scala xxxx In Depth where xxxx is a concept like Extractor).