Workshop on Play and Akka (Verizon VDSI)

Nirmalya Sengupta

Customary self-sell

- (very) old hat
- C -> C++ -> Java -> Scala (bash/perl)
- Only Unix/Linux, No Microsoft! :-)
- Working as a freelance backend stack developer/mentor/architect
- Integration framework (CORBA), telecom, multiplayer gaming, rule engine, streaming analytics
- India, Ireland, Germany
- Training / Consulting whenever it interests me

Sessions: a look ahead

- A brief detour of some JDK8 features
- Get to know Akka and Actors: the story
- Get to know Play framework: separation of M,V and C
- Put to practise, our understanding, during the journey
- Being Reactive: inquisitive look at Akka Http / Streaming

Rules of the game

- Overall concepts using slides
- Whiteboard explanations
- Download code snippets, then modify (dirtying our hands), then **write more**
- Get comfortable with tools
- Questions: any time | Answers: best effort (I may not know, and you may)
- Learning: everyone (I, and you all)
- Enjoy, reflect, look for ***Aha*** moments

Your machines: ready?

- JDK 8 (not JRE)
- Eclipse
- Activator
- Git (client)
- sbt
- cURL
- browser

Download the following using git:

https://github.com/nsengupta/Akka-Lab-Projects

JDK 8: a bunch of timely offerings

- Language
  - Lambda Expressions,Method References,Generics etc.
- Security
- Tools
  - jjs
- Scripting
- Concurrency
- etc.

JDK 8: Autoboxing

- Create Objects from fundamental datatypes
  - Fundamental to Objectified: AutoBoxing
  - Objectified to Fundamental: AutoUnboxing
- Necessary for Collections
- JDK defines the autoboxing rules

JDK 8: Generics

- Available since Java 5
- Think: type of types
    - An Array of Integers
    - A Map of Strings and Persons (*pojo*)
- Advantage: Compile time checking
- Necessary for Collections

JDK 8: Functional Interface

- Interfaces having a single abstract method
- The type of the method and its parameters are determinable
- @FunctionalInterface annotation available
- Java 8 provides a number of them
  - java.util.function.*
- Helps in writing concise, readable code
- Candidates for lambda expressions

JDK 8: Lambda Expressions

- Handy unnamed functions
- No explicit object creation
- Uses Functional Interface
- Compiler helps in attributing the right type
- Can capture variables from its enclosing scope
- Use for small, direct, obvious cases

JDK 8: Method/Constructor References

- Lambda Expressions can use existing methods
  - We refer to them by **name**, as if they are parameters
- The are four different types available
  - Static method name -> **containingClass :: methodName**
  - instance method name -> **containingInstance :: methodName**
  - Arbitrary object -> **ClassName :: methodName**
  - Constructor -> **ClassName :: new**

JDK 8: Streams

- Traversal of Collections, but not behaving like Iterators
- Functional access to elements of a Collection
- Streams have no storage and no modification
- Lazy in nature
- 2 types of operations
  - Intermediate - produces streams
  - Terminal - prduces values
- Consumable (like Iterators)