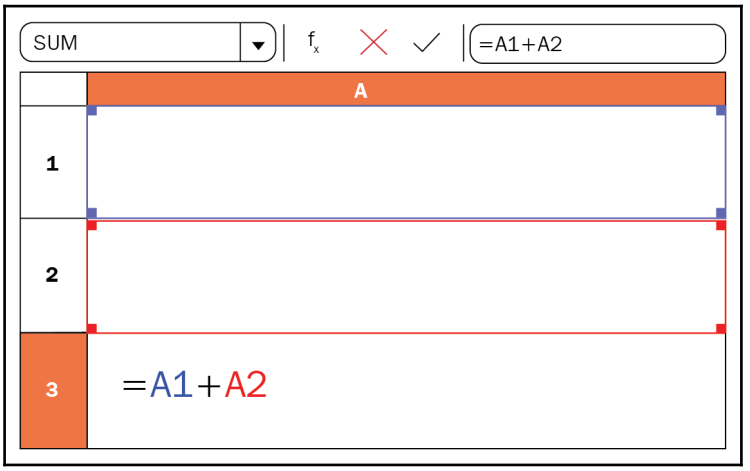
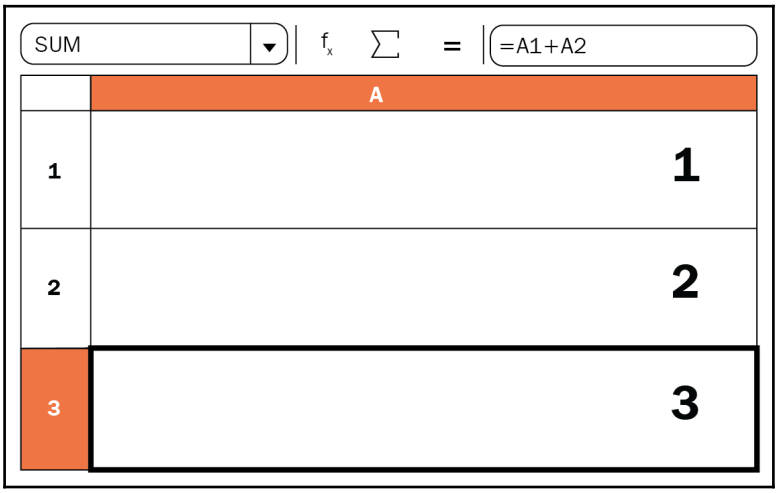
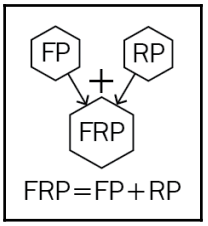
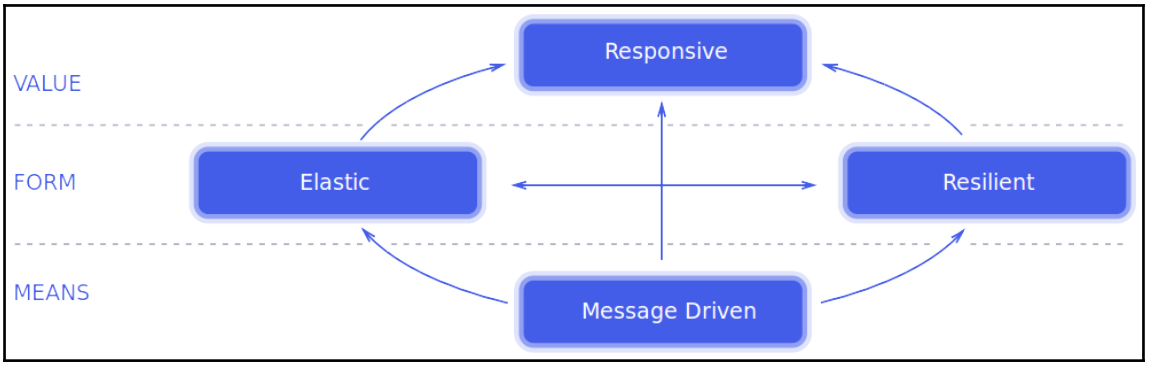
Chapter 1: Getting Started with Reactive and Functional Programming

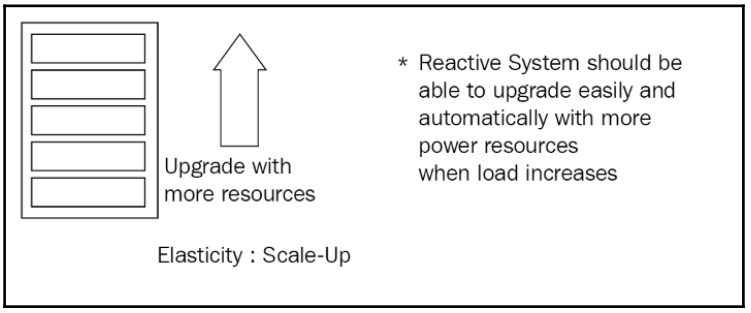


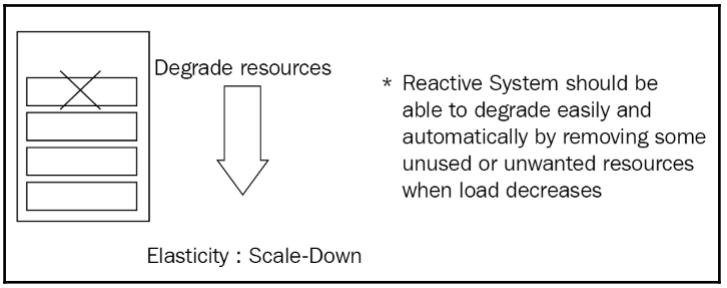


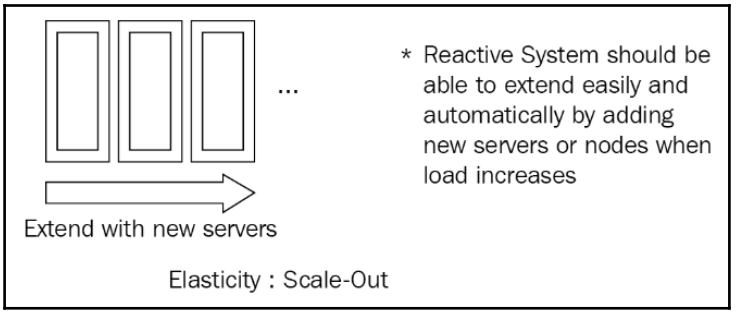

Publisher
E_n . . . E_2 E_1
subscriber
Events Stream

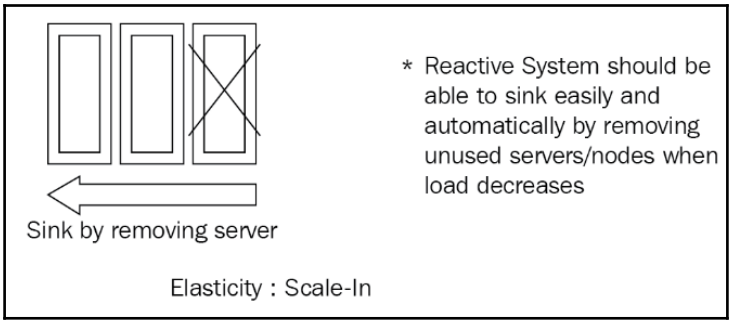

FP
RP
FRP
FRP=FP+RP

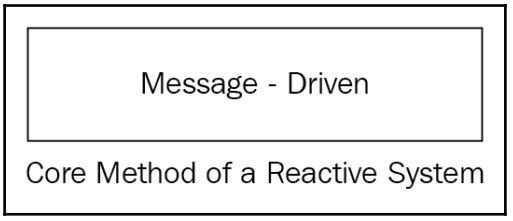

VALUE
FORM
MEANS
Responsive
Elastic
Resilient
Message Driven

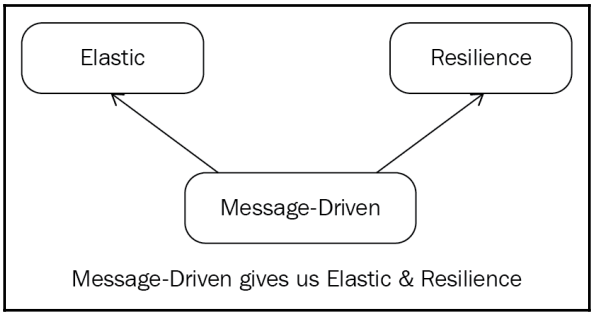

Upgrade with
more resources
Elasticity : Scale-Up
* Reactive System should be able to upgrade easily and automatically with more power resources when load increases

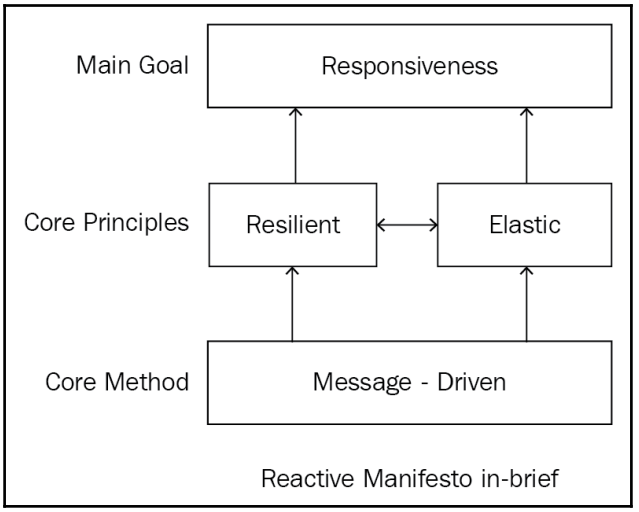

Degrade resources
* Reactive System should be able to degrade easily and automatically by removing some unused or unwanted resources when load decreases
Elasticity : Scale-Down

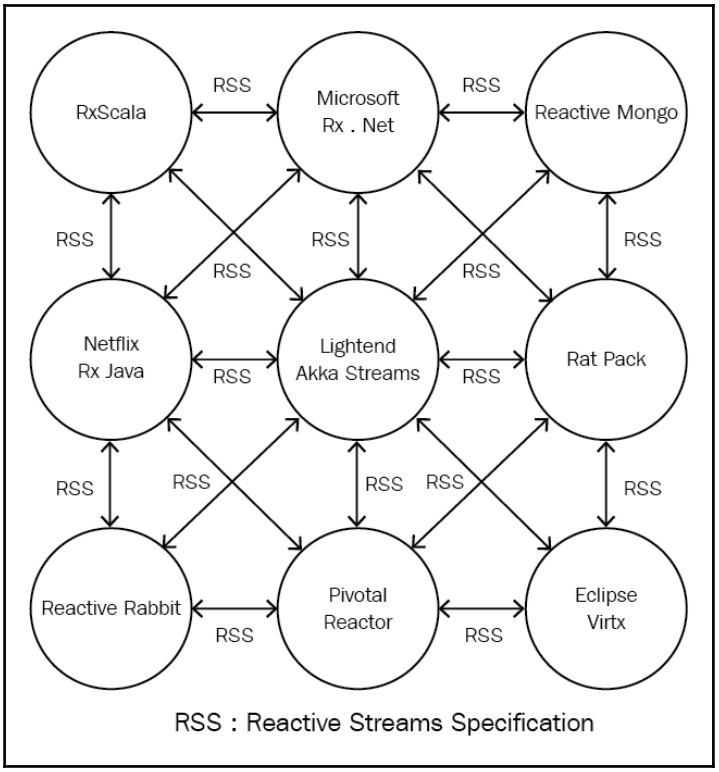

...
* Reactive System should be able to extend easily and automatically by adding new servers or nodes when load increases
Extend with new servers
Elasticity : Scale-Out

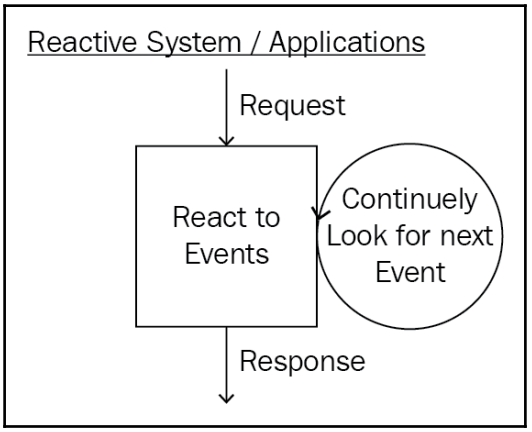

* Reactive System should be able to sink easily and automatically by removing unused servers/nodes when load decreases
Sink by removing server
Elasticity : Scale-In

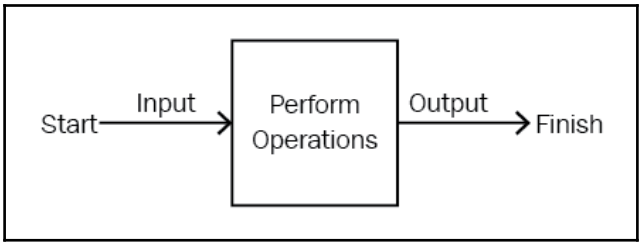

Message - Driven
Core Method of a Reactive System

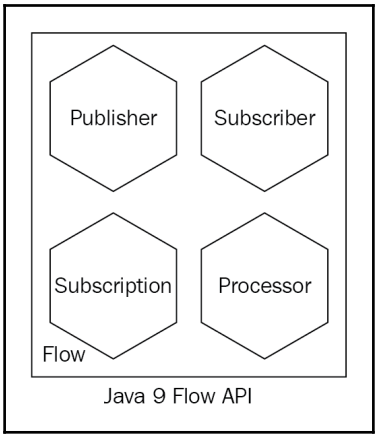

Elastic
Resilience
Message-Driven
Message-Driven gives us Elastic & Resilience

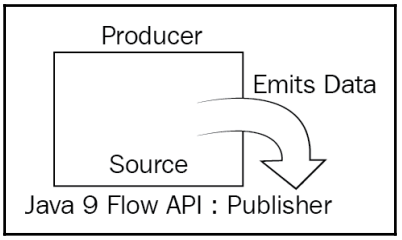

Main Goal
Responsiveness
Core Principles
Resilient
Elastic
Core Method
Message - Driven
Reactive Manifesto in-brief

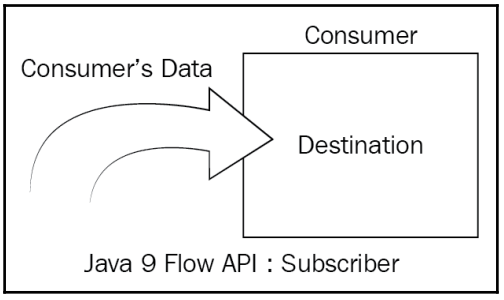

RxScala
Microsoft
Rx . Net
Reactive Mongo
Netflix
Rx Java
Lightend
Akka Streams
Rat Pack
Reactive Rabbit
Pivotal
Reactor
Eclipse
Virtx
RSS
RSS : Reactive Streams Specification

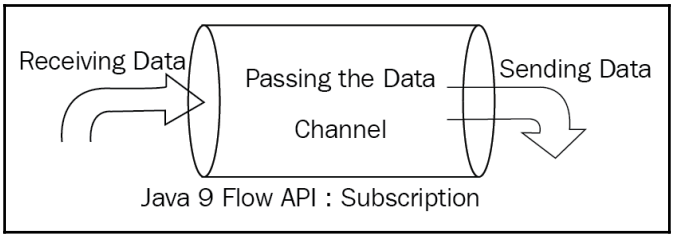

Reactive System / Applications
Request
React to
Events
Continuely
Look for next
Event
Response


Start
Input
Perform Operations
Output
Finish

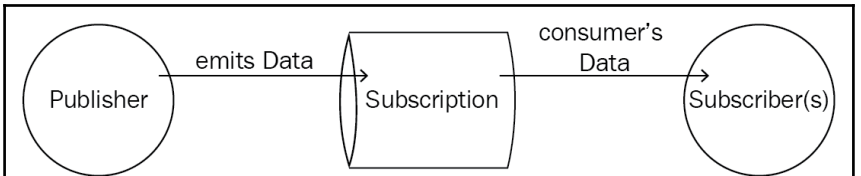

Publisher
Subscriber
Subscription
Processor
Flow
Java 9 Flow API


Producer
Emits Data
Source
Java 9 Flow API : Publisher

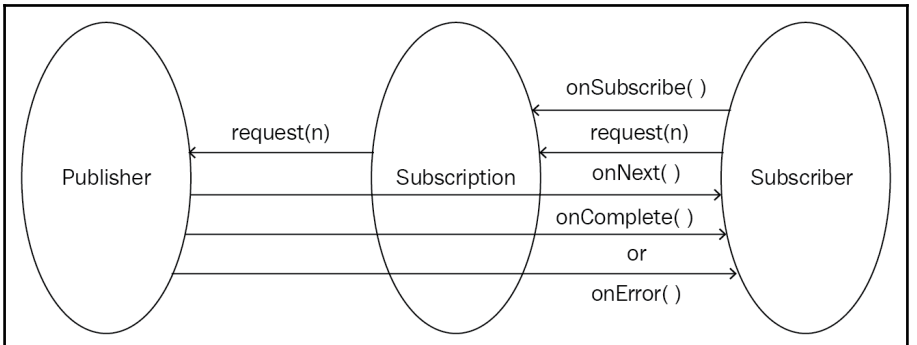

Consumer
Consumer's Data
Destination
Java 9 Flow API : Subscriber



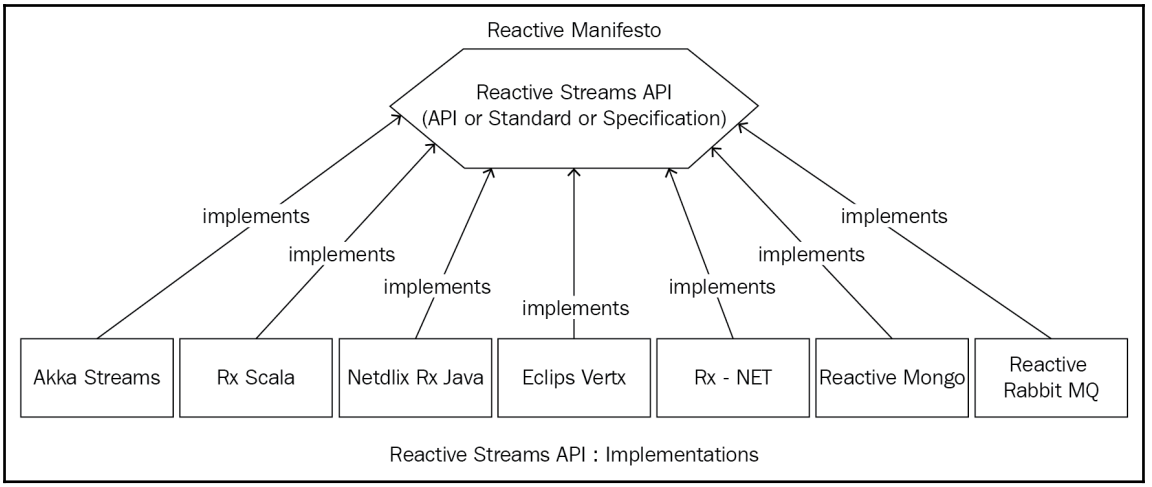
Java 9 Flow API : Subscription

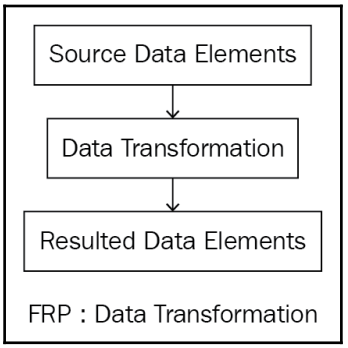


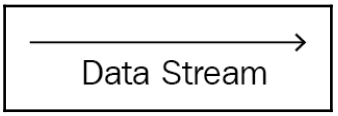
Relationship between Java9 Flow API Components

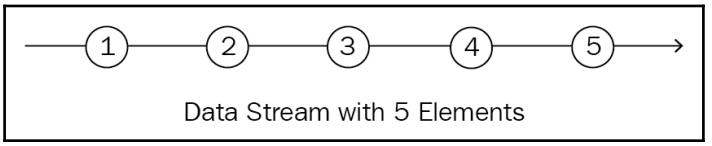


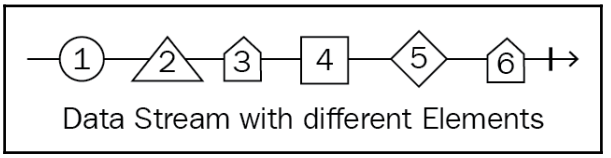
Java 9 Flow API Operations

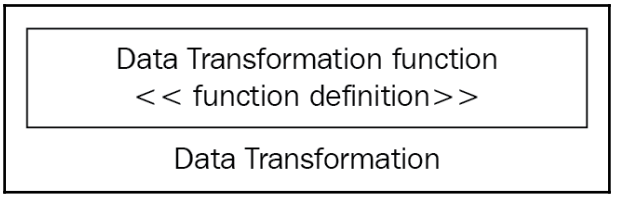

Reactive Manifesto
Reactive Streams API
(API or Standard or Specification)
implements
implements
implements
implements
implements
implements
implements
Akka Streams
Rx Scala
Netdlix Rx Java
Eclips Vertx
Rx - NET
Reactive Mongo
Reactive Rabbit MQ
Reactive Streams API : Implementations

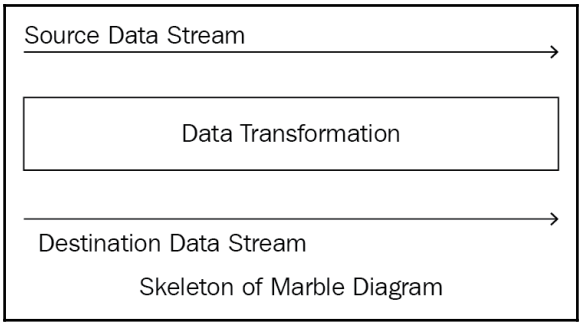

Source Data Elements
Data Transformation
Resulted Data Elements
FRP : Data Transformation

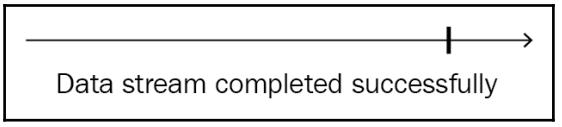

Data Stream

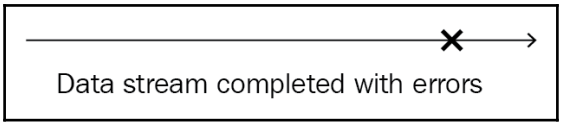

1
2
3
4
5
Data Stream with 5 Elements

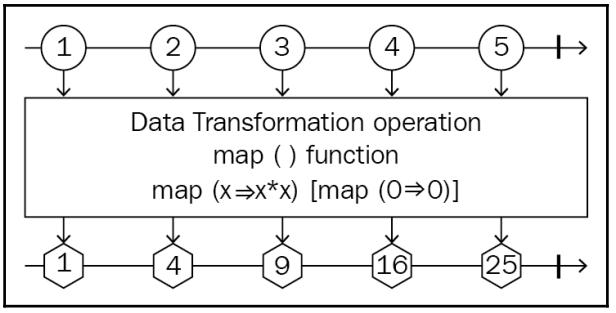

1
2
3
4
5
6
Data Stream with different Elements

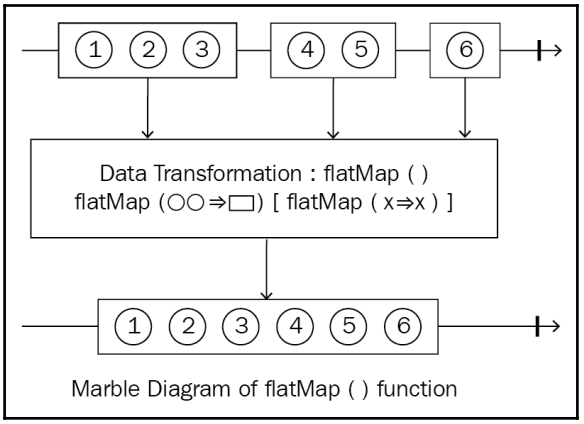

Data Transformation function
<< function definition>>
Data Transformation


Source Data Stream
Data Transformation
Destination Data Stream
Skeleton of Marble Diagram

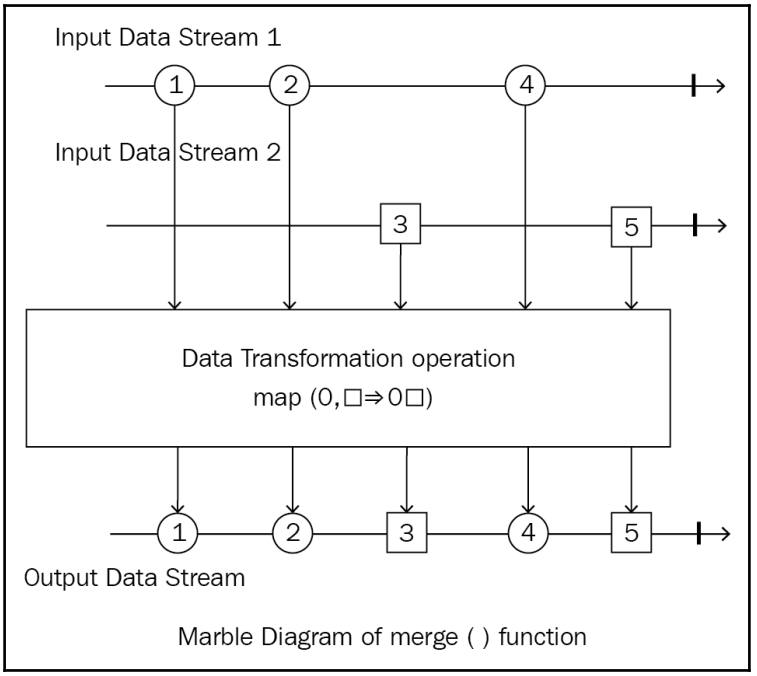

Data stream completed successfully


Data stream completed with errors

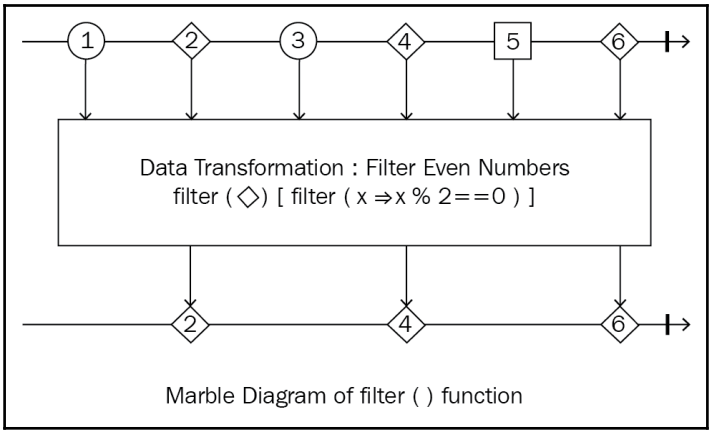

1
2
3
4
5
Data Transformation operation
map ( ) function
map (x⇒x*x) [map (0⇒0)]
1
4
9
16
25



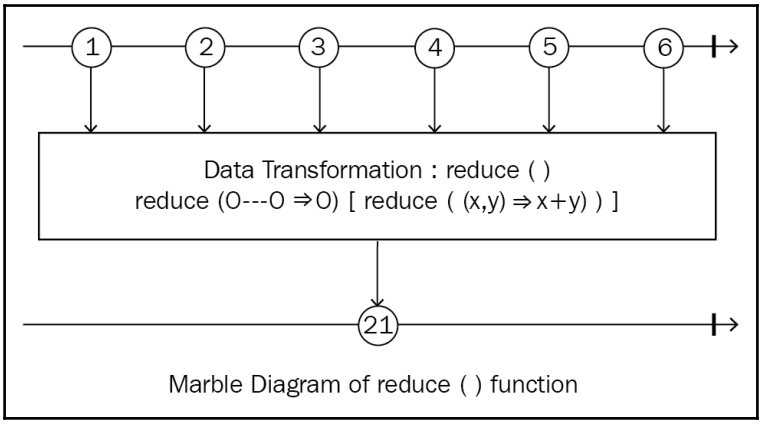
Marble Diagram of flatMap ( ) function



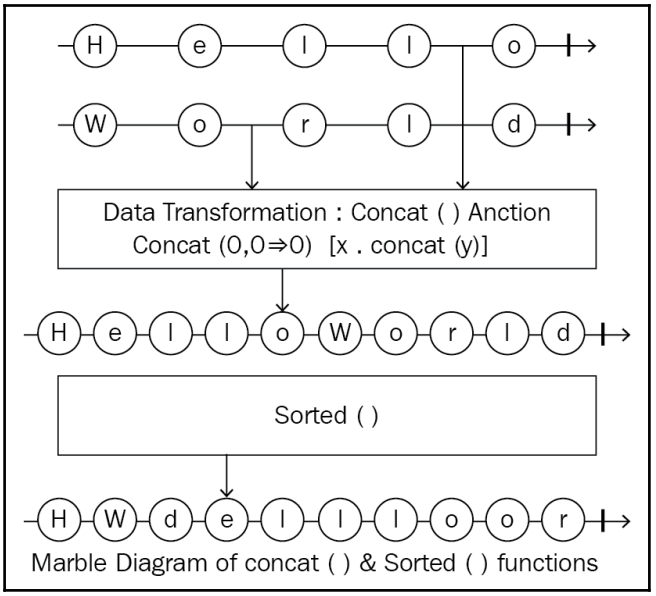
Marble Diagram of merge ( ) function



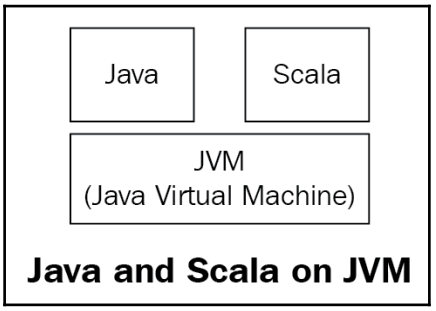
Marble Diagram of filter ( ) function

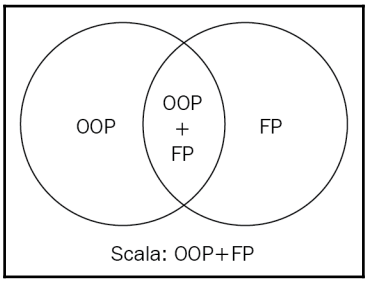


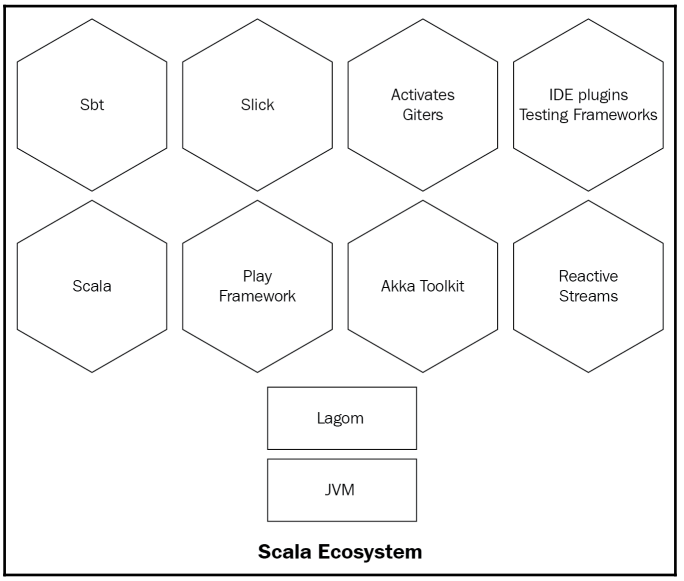
Marble Diagram of reduce ( ) function


H e l l o
W o r l d
Data Transformation : Concat ( ) Anction
Concat (0,0⇒0) [x . concat (y)]
H e l l o W o r l d
Sorted ( )
H W d e l l l o o r
Marble Diagram of concat ( ) & Sorted ( ) functions

Chapter 2: Functional Scala



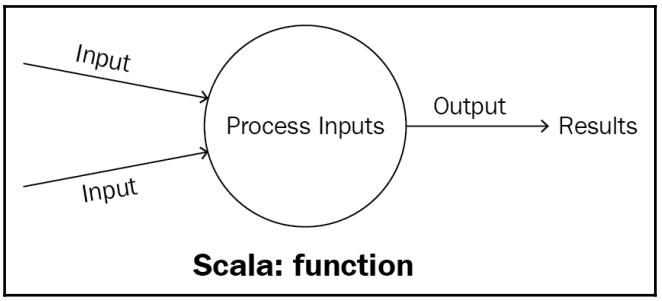


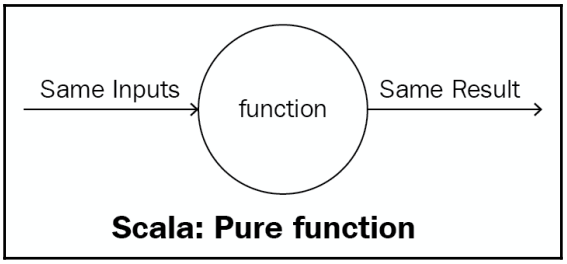


```
rambabuposa@ram: ~
rambabuposa@ram:~$ scala
Welcome to Scala 2.12.3 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_121).
Type in expressions for evaluation. Or try :help.

scala>
```

Scala FP Main Features

- ✓ Immutability
- ✓ Functions as First-class Citizens
- ✓ No side effects
- ✓ Pure Functions
- ✓ RT (Referential Transparency)





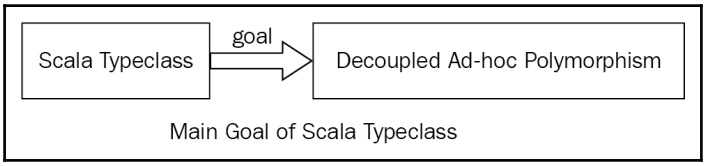
(Inputs) ⟹ Output

Scala Anonymous Function Syntax

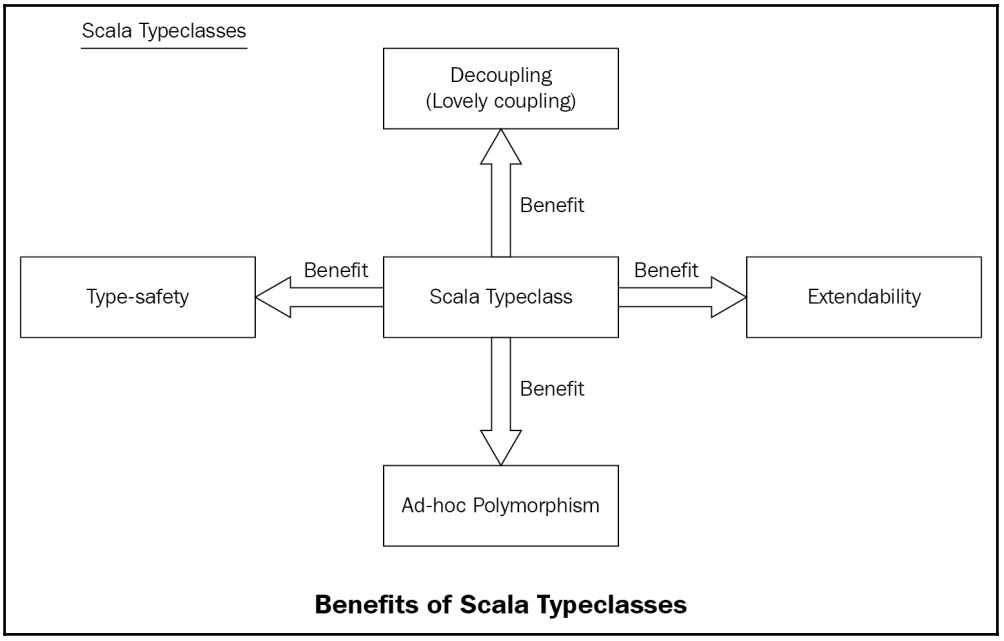
Anonymous Function: Inputs ⟹ Output

Function Inputs	Function Output	Anonymous Function	Anonymous Function Type
(Int)	(Int)	(Int) ⟹ Int	funtion1
(Int, Int)	(Int)	(Int, Int) ⟹ Int	function2
(Int, Int, Int)	(Int)	(Int, Int, Int) ⟹ Int	function3
. . .			
(Int, Int, ... Int) 22 Parameters	(Int)	(Int, Int, ... Int) ⟹ Int	function22

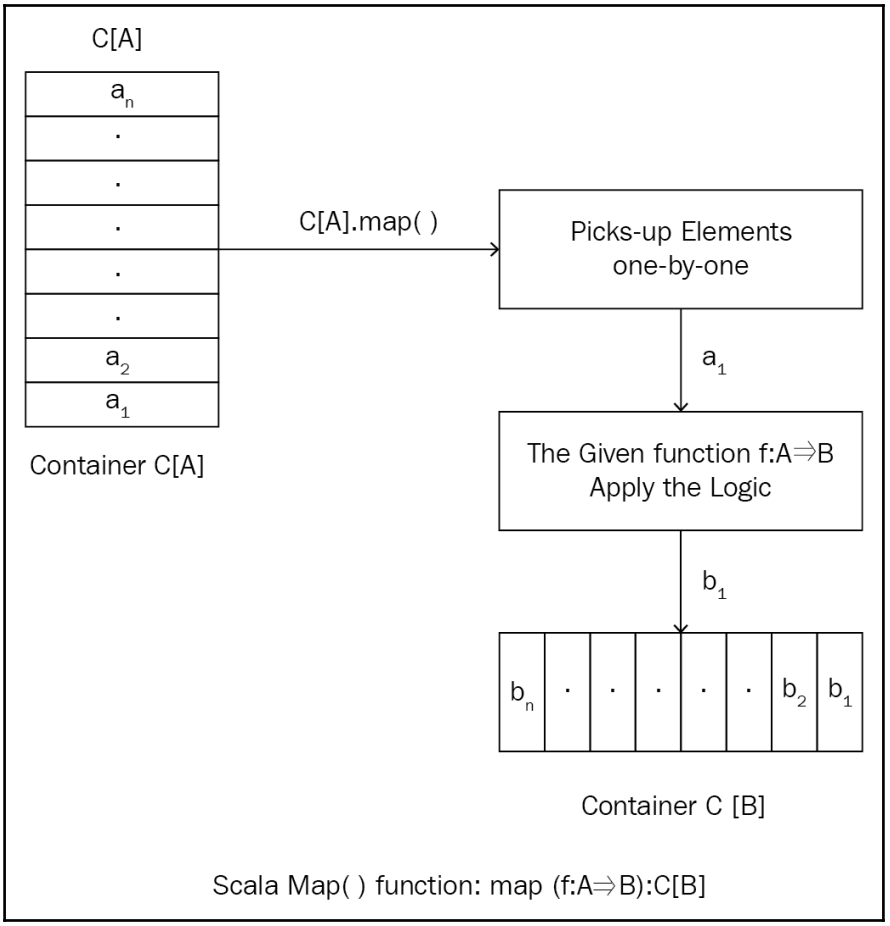
Scala Anonymous function

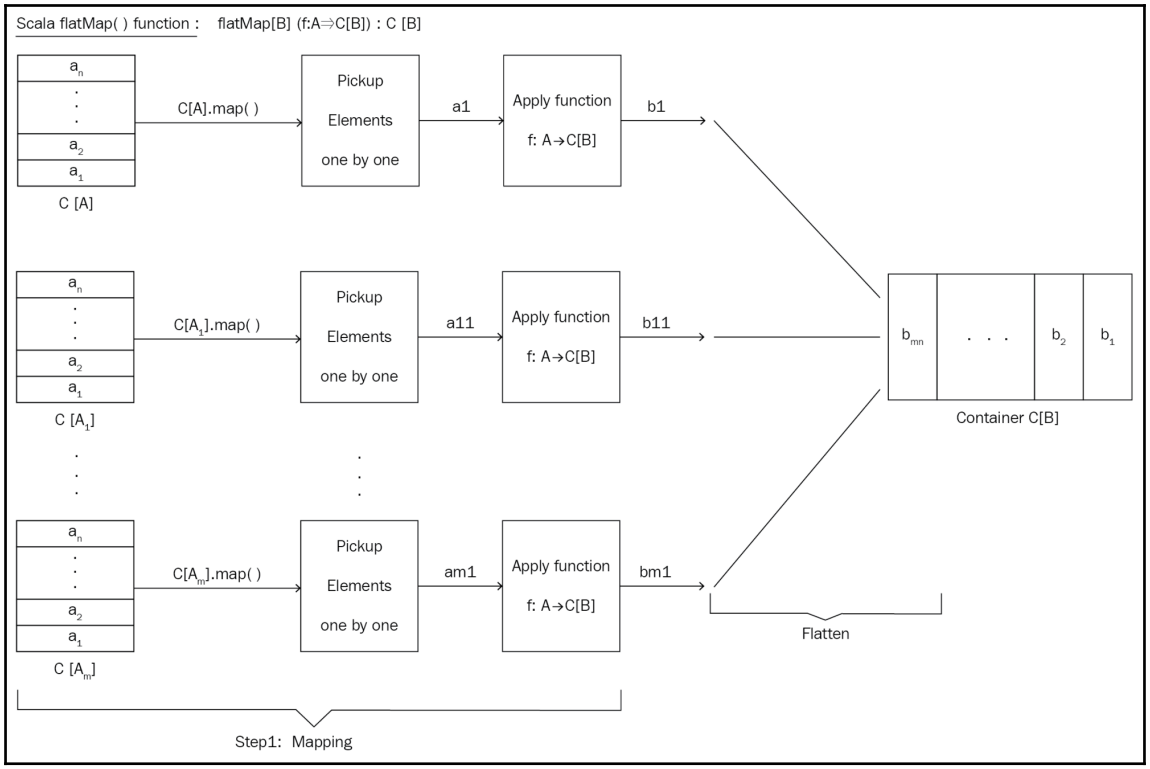


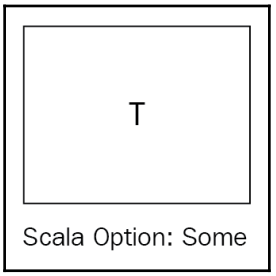
Main Goal of Scala Typeclass

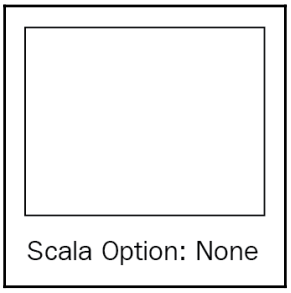


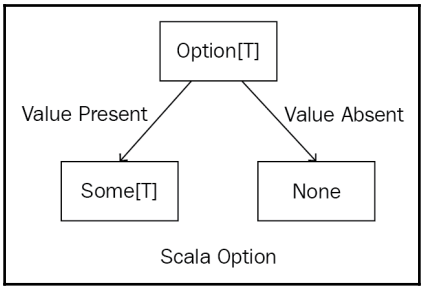
Benefits of Scala Typeclasses

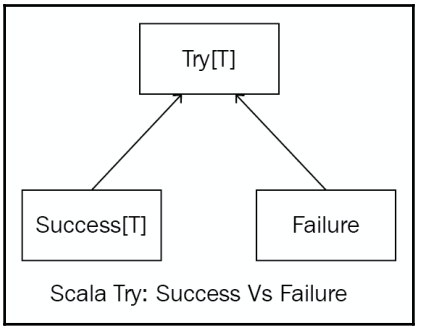

C[A]
a_n
.
.
.
.
.
a_2
a_1
Container C[A]
C[A].map( )
Picks-up Elements
one-by-one
a_1
The Given function f:A⇒B
Apply the Logic
b_1
b_n . . . . . b_2 b_1
Container C [B]
Scala Map( ) function: map (f:A⇒B):C[B]

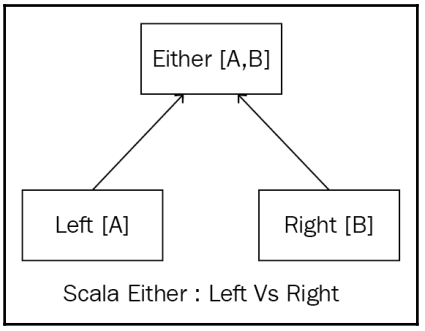

Scala flatMap( ) function :
flatMap[B] (f:A⇒C[B]) : C [B]
C [A]
C[A].map( )
Pickup Elements one by one
a1
Apply function f: A→C[B]
b1
C [A1]
C[A1].map( )
a11
b11
C [Am]
C[Am].map( )
am1
bm1
Container C[B]
Flatten
Step1: Mapping


T
Scala Option: Some

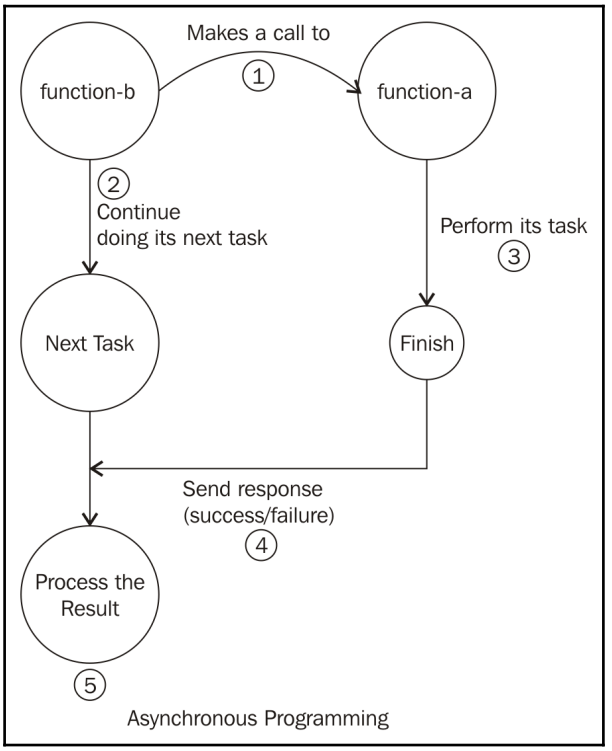

Scala Option: None

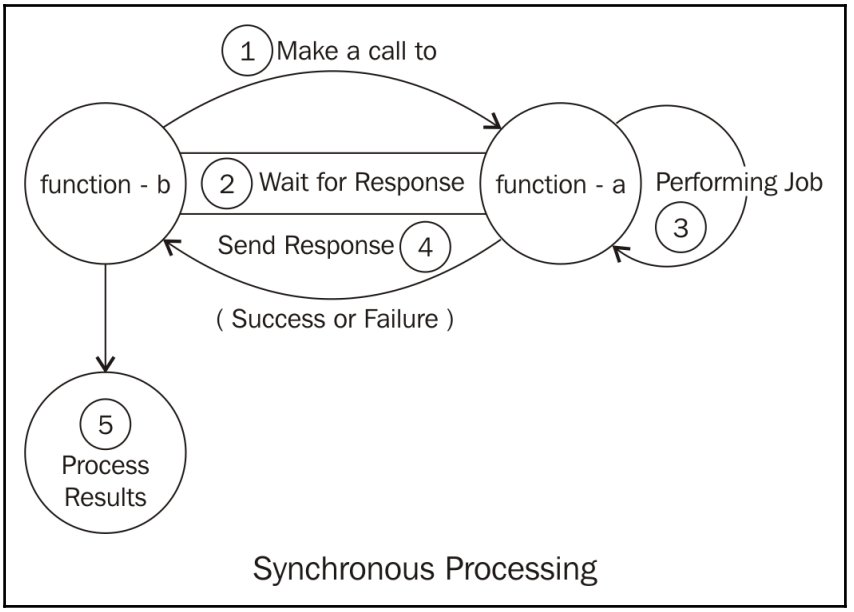

Option[T]
Value Present
Value Absent
Some[T]
None
Scala Option

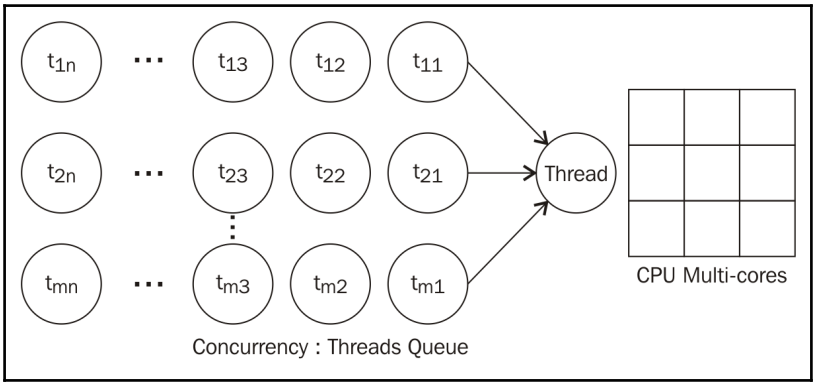

Try[T]
Success[T]
Failure
Scala Try: Success Vs Failure

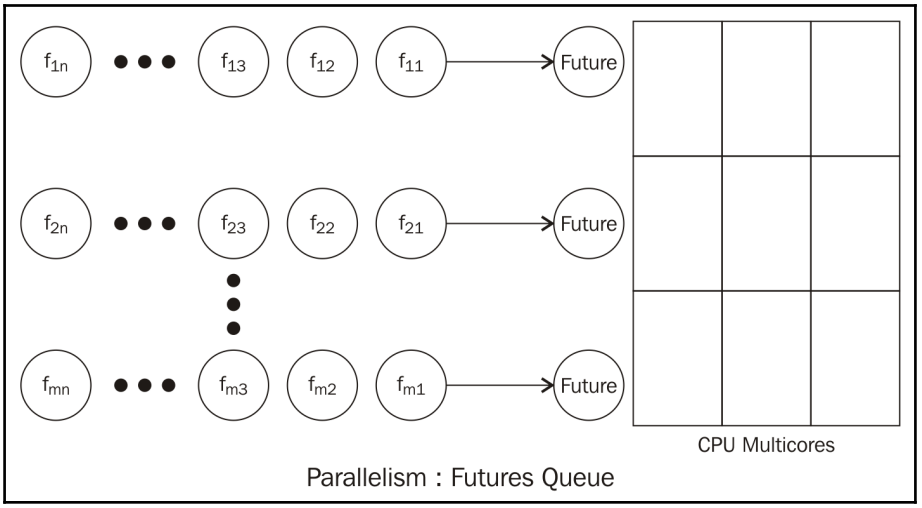

Either [A,B]
Left [A]
Right [B]
Scala Either : Left Vs Right

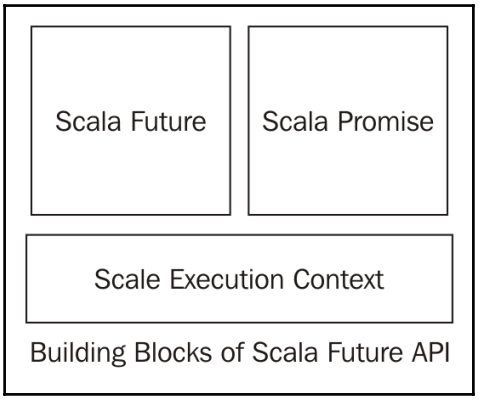
Chapter 3: Asynchronous Programming with Scala

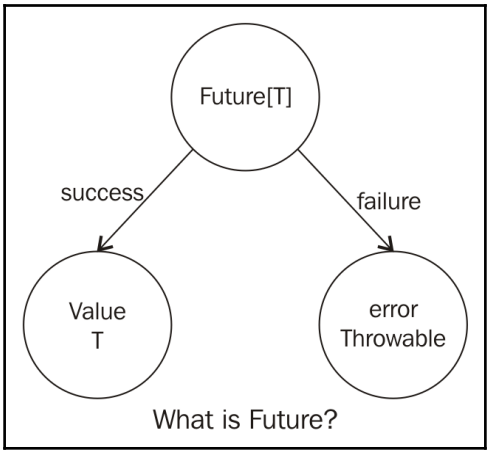


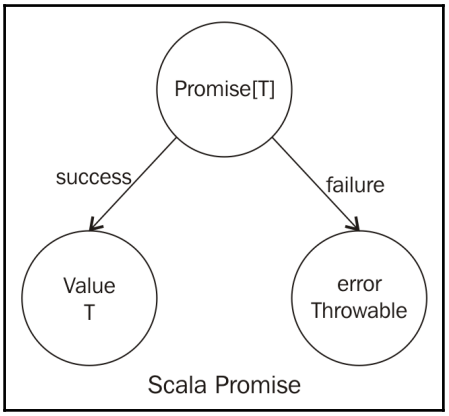

1 Make a call to
function - b
2 Wait for Response
function - a
Performing Job
3
Send Response 4
(Success or Failure)
5
Process
Results
Synchronous Processing

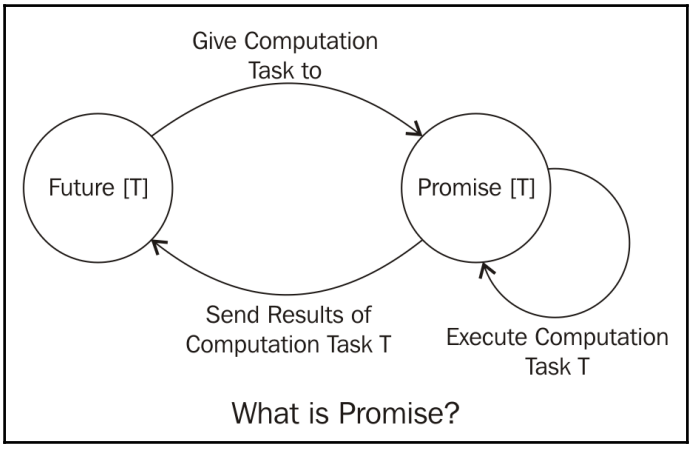

t1n
t13
t12
t11
t2n
t23
t22
t21
Thread
tmn
tm3
tm2
tm1
CPU Multi-cores
Concurrency : Threads Queue

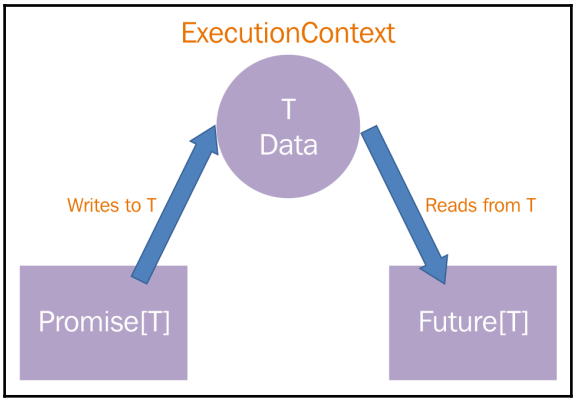

f1n
f13
f12
f11
Future
f2n
f23
f22
f21
Future
fmn
fm3
fm2
fm1
Future
CPU Multicores
Parallelism : Futures Queue

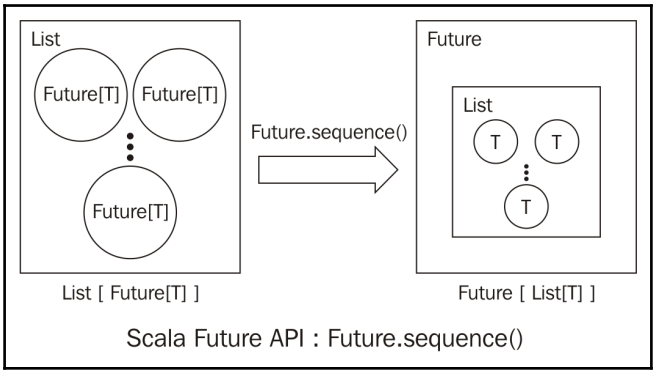

Scala Future
Scala Promise
Scale Execution Context
Building Blocks of Scala Future API


Future[T]
success
failure
Value
T
error
Throwable
What is Future?

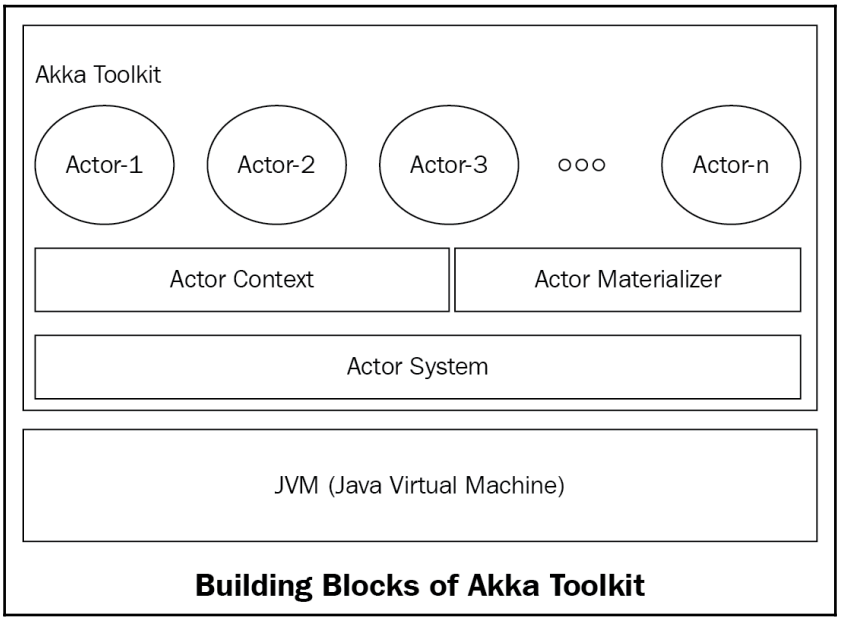

Promise[T]
success
failure
Value
T
error
Throwable
Scala Promise


Give Computation
Task to
Future [T]
Promise [T]
Send Results of
Computation Task T
Execute Computation
Task T
What is Promise?

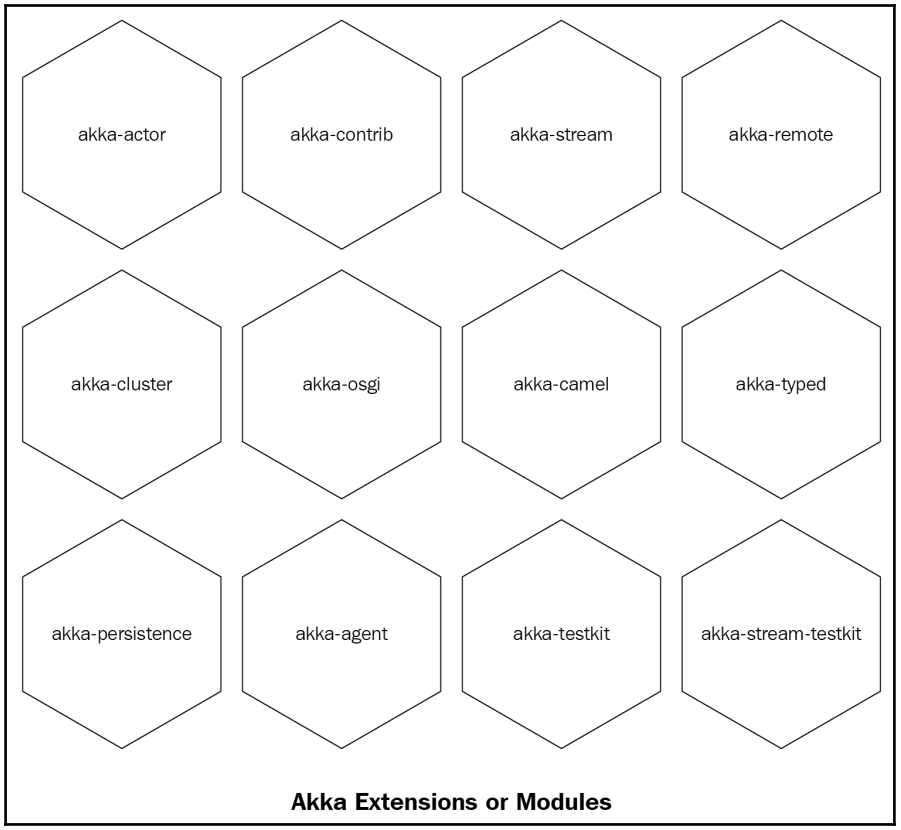

ExecutionContext
T
Data
Writes to T
Reads from T
Promise[T]
Future[T]


List
Future[T]
Future[T]
Future[T]
Future.sequence()
Future
List
T
T
T
List [Future[T]]
Future [List[T]]
Scala Future API : Future.sequence()

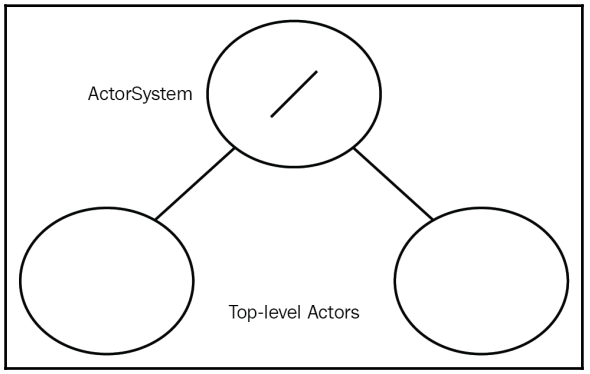
Chapter 4: Building Reactive Applications with Akka

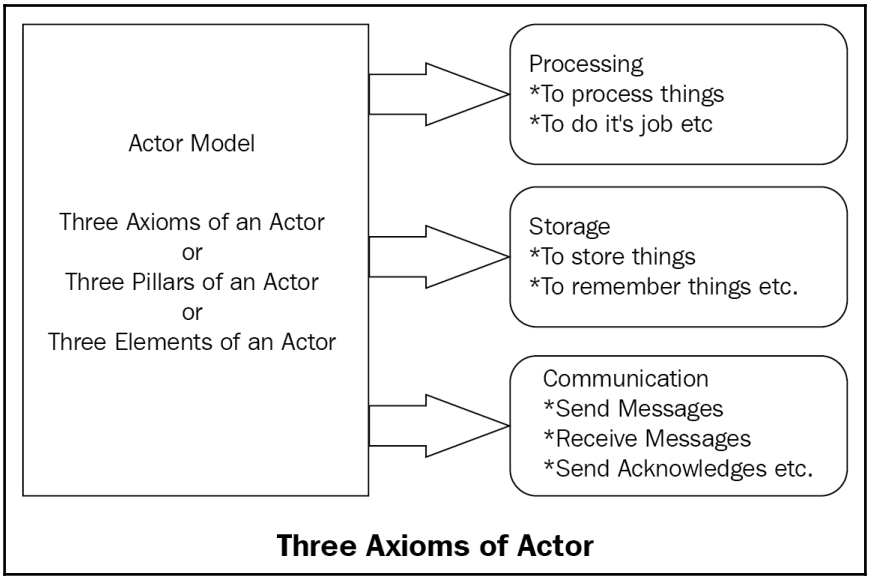


Building Blocks of Akka Toolkit

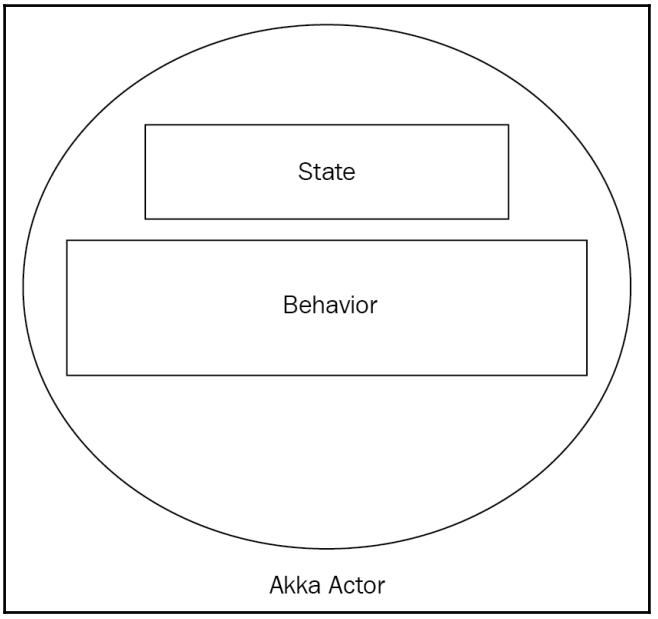


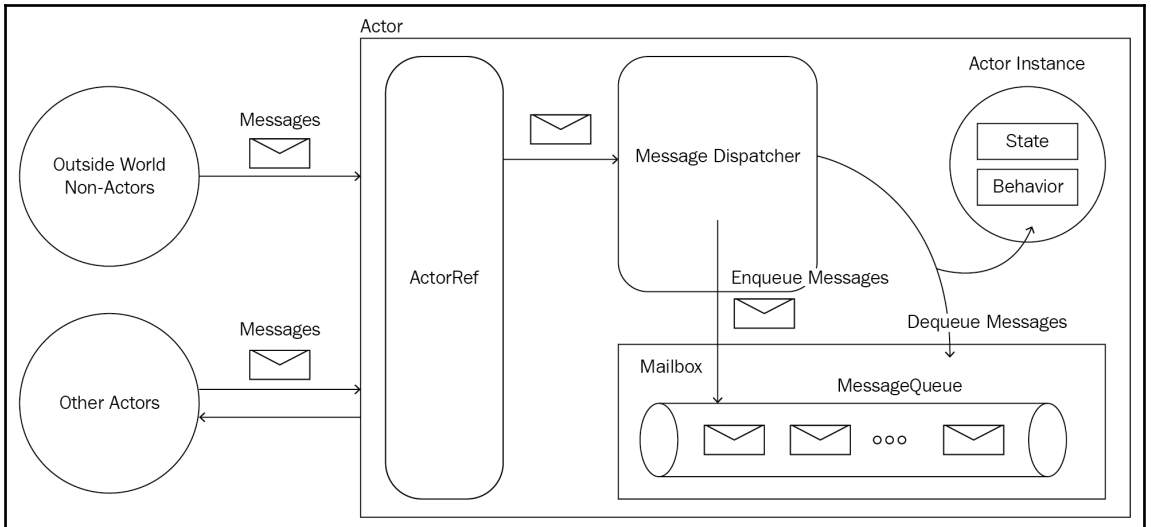
Akka Extensions or Modules



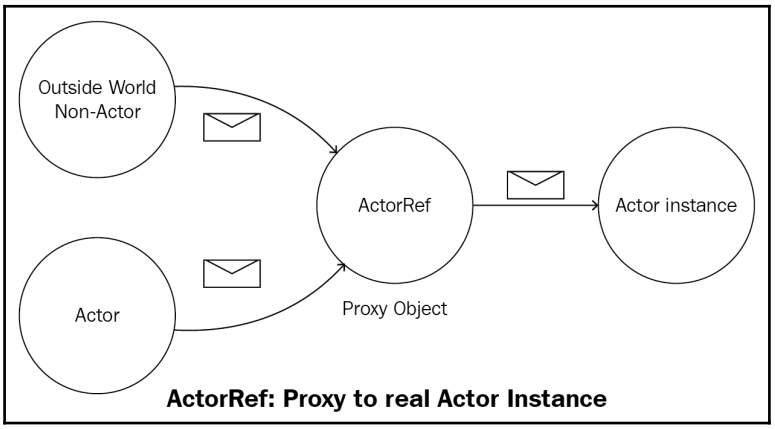


Three Axioms of Actor

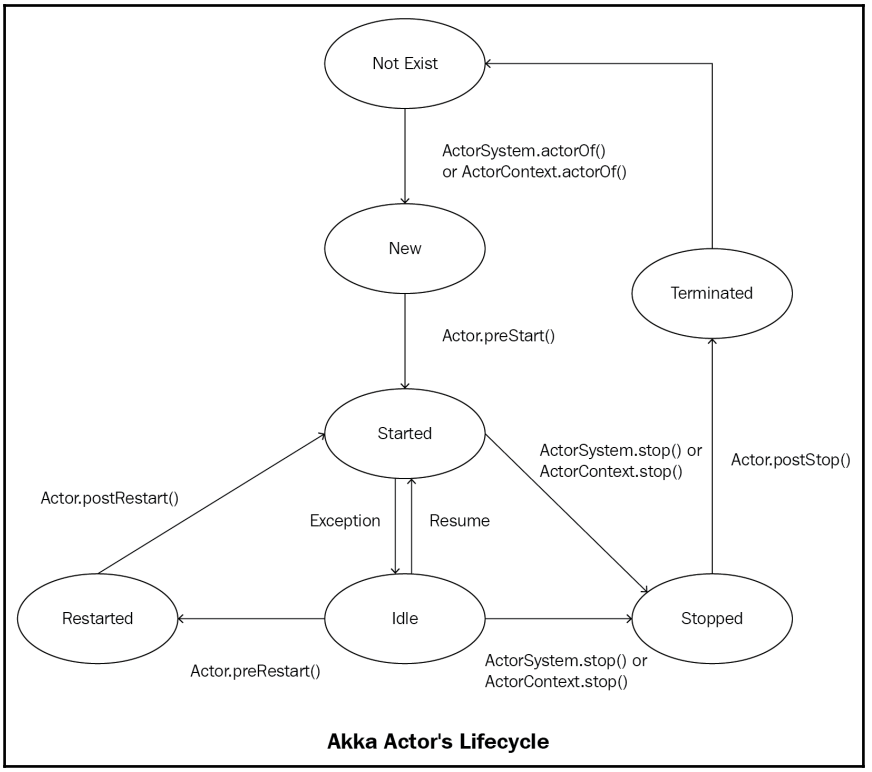




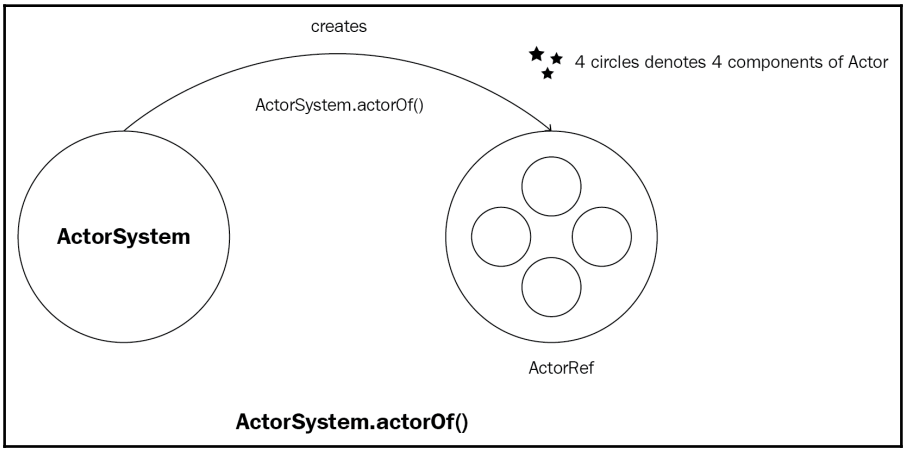
Internal Components of Akka Actor



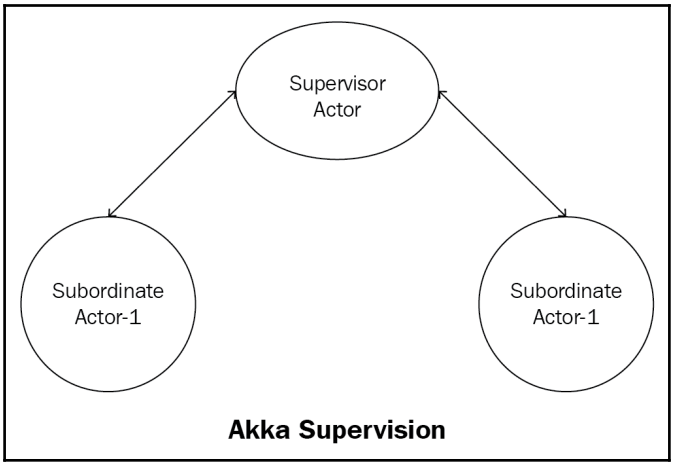
ActorRef: Proxy to real Actor Instance



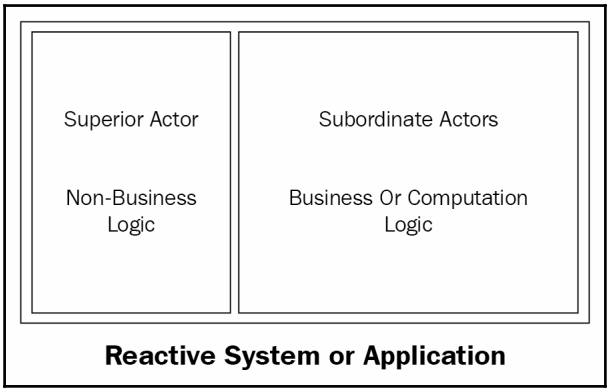
Akka Actor's Lifecycle



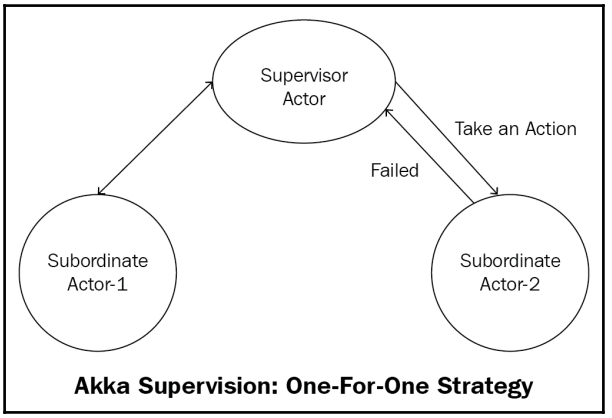
ActorSystem.actorOf()



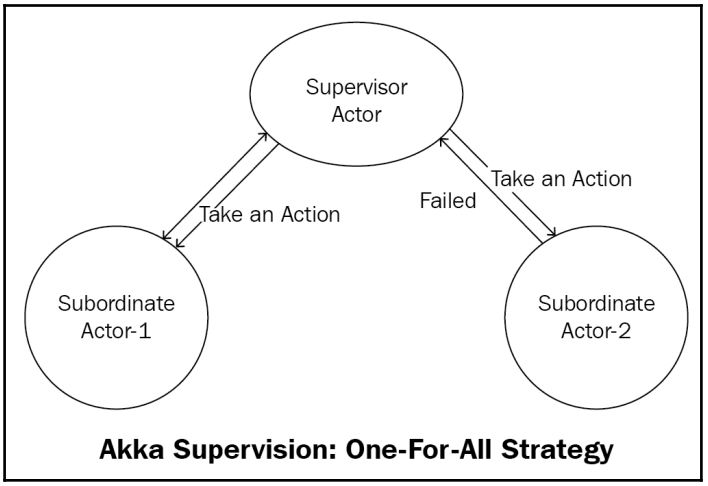
Akka Supervision



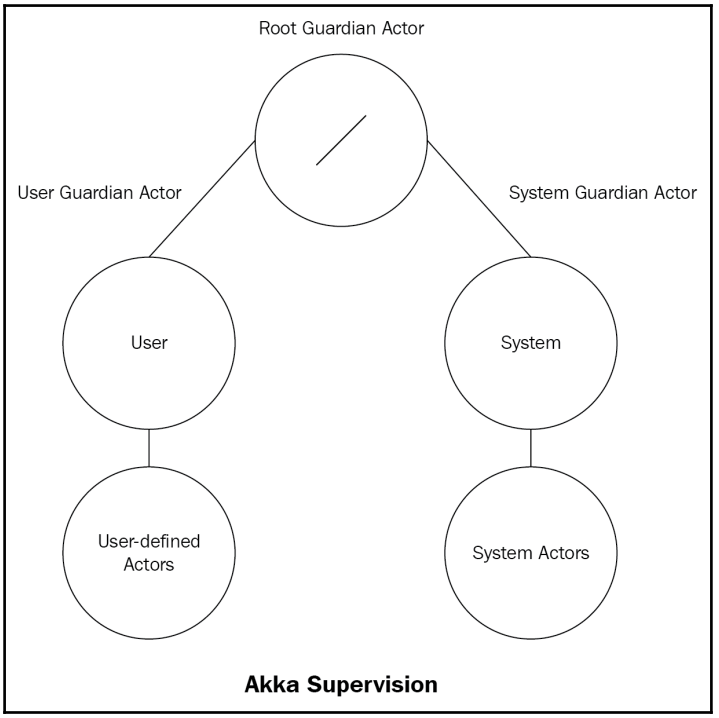
Reactive System or Application



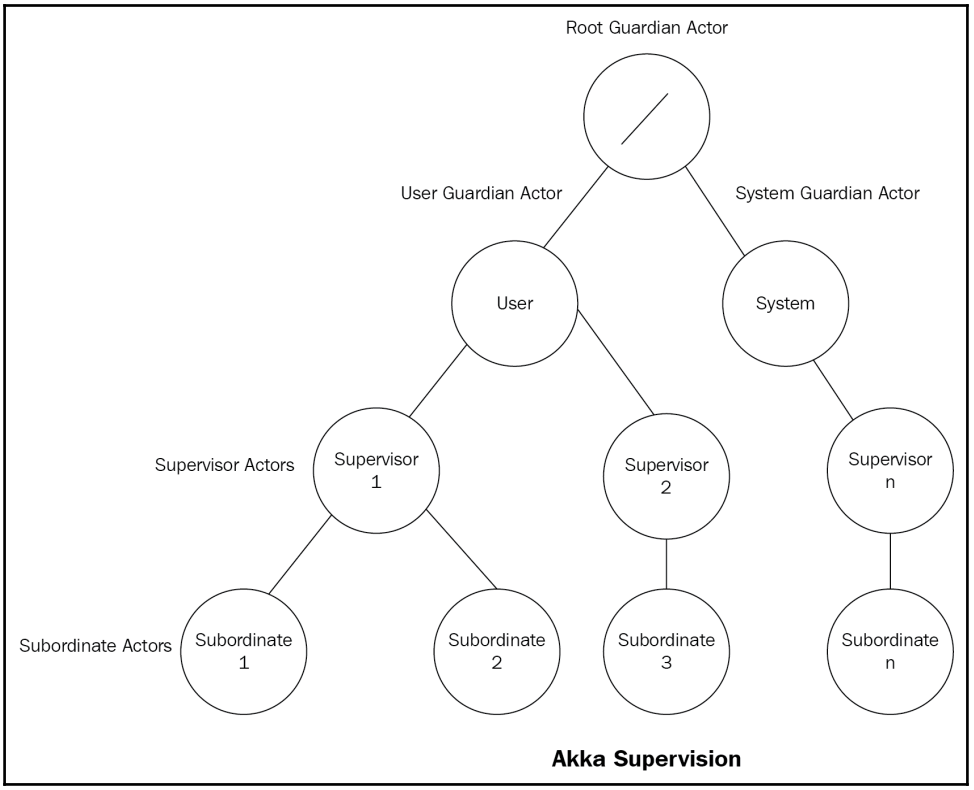
Akka Supervision: One-For-One Strategy



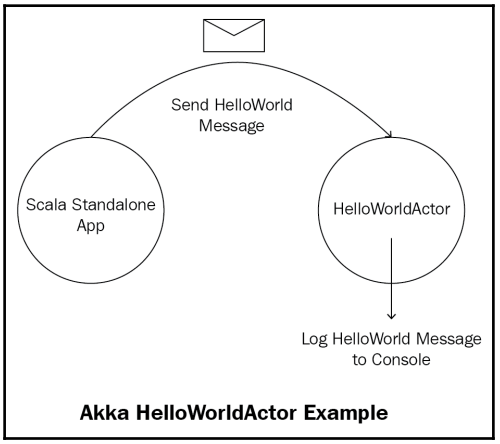
Akka Supervision: One-For-All Strategy



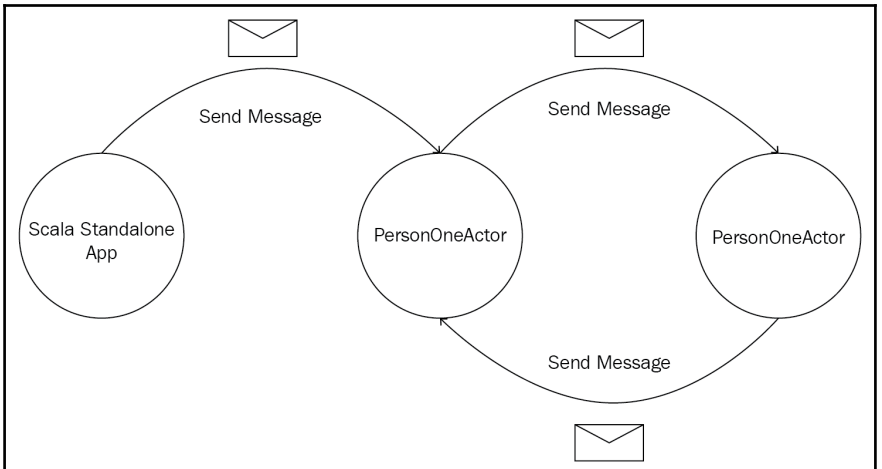
Akka Supervision



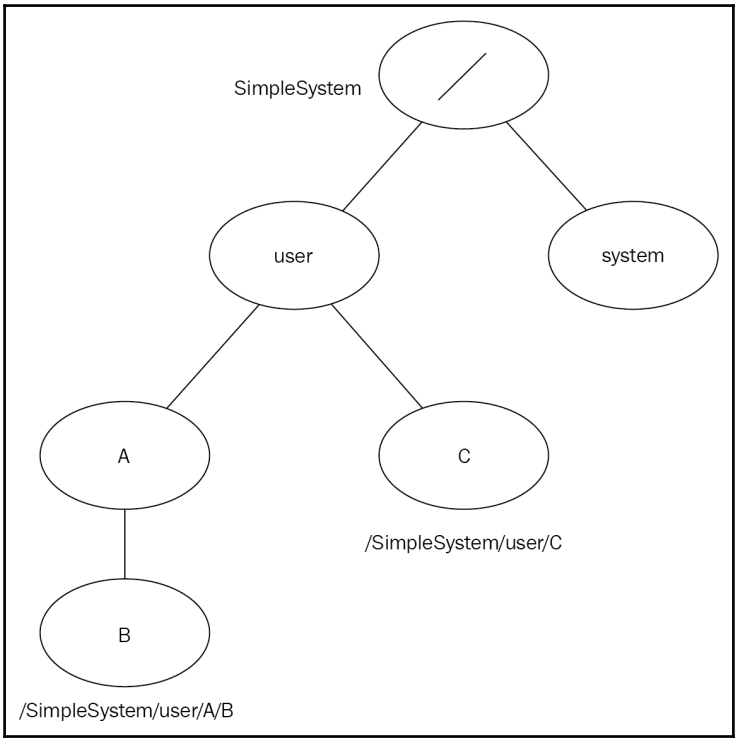
Akka Supervision

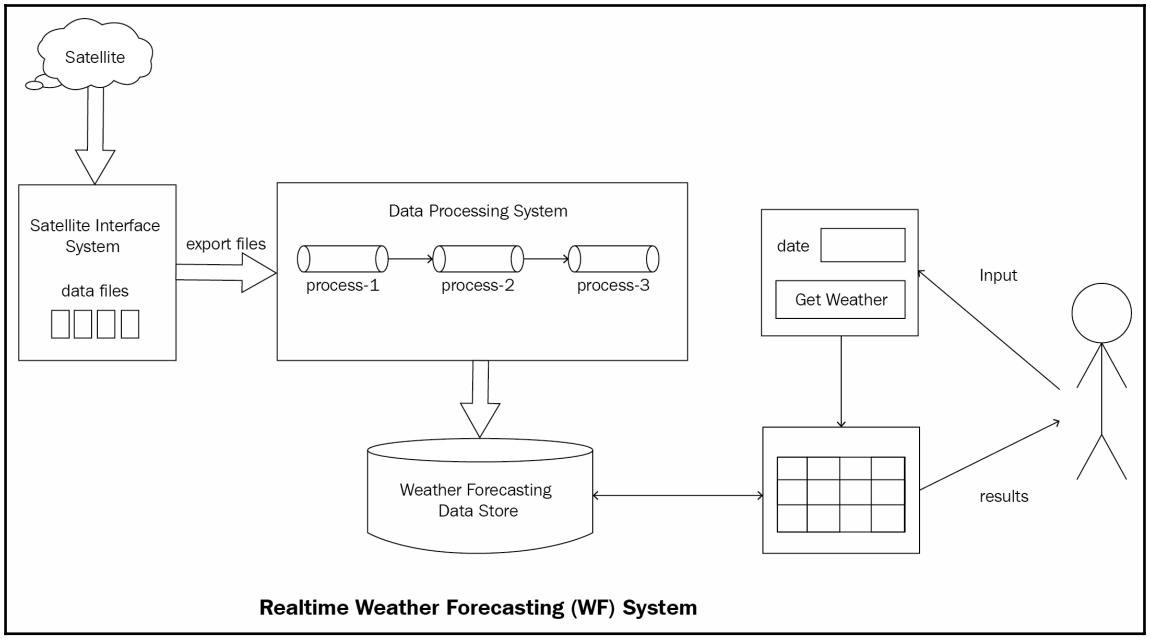


Akka HelloWorldActor Example

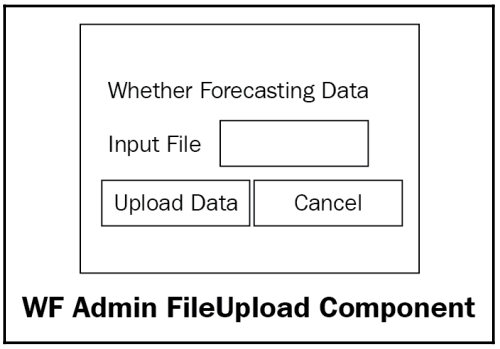


Akka Actors Communication Example

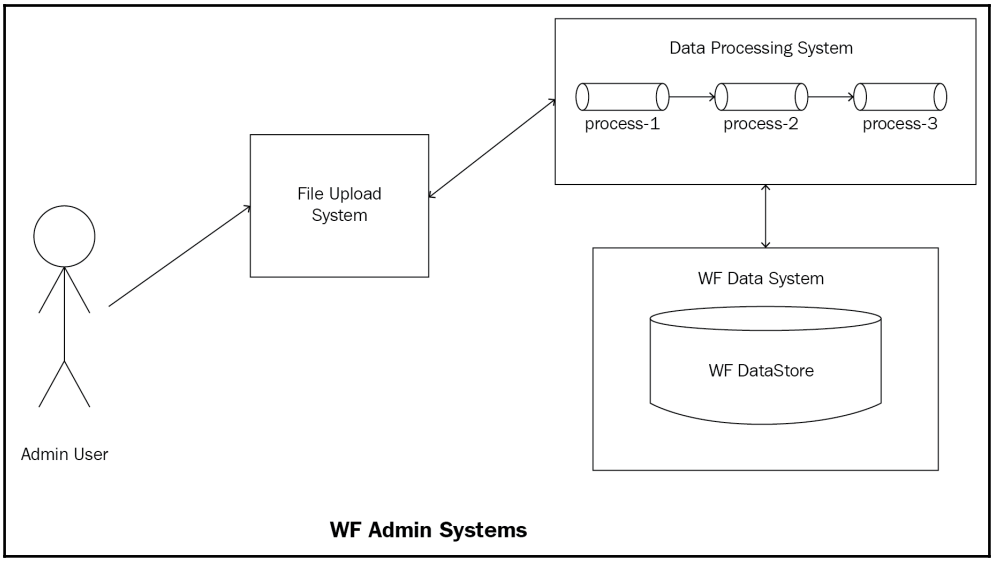




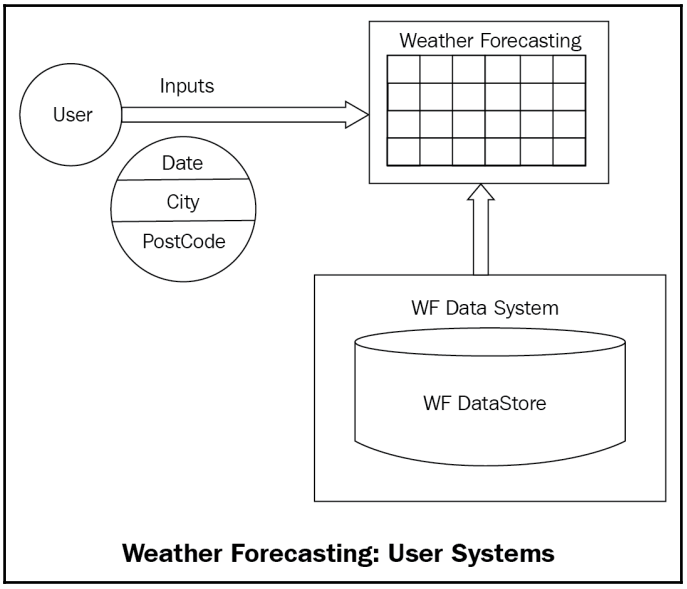
Realtime Weather Forecasting (WF) System



WF Admin FileUpload Component

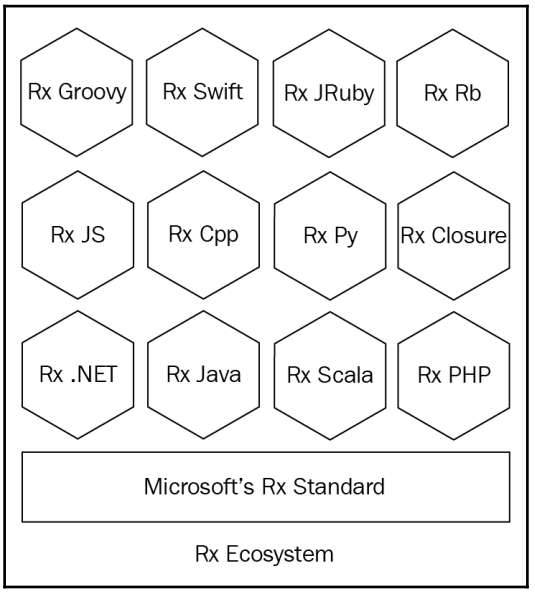


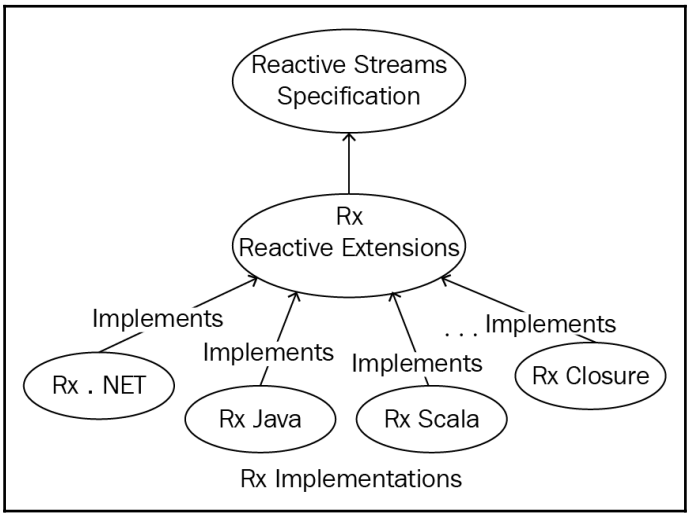
WF Admin Systems

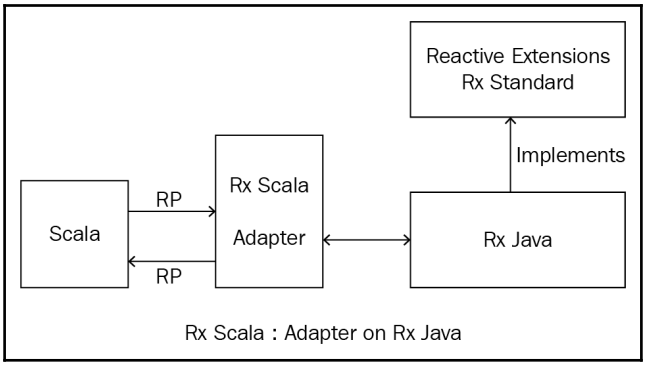


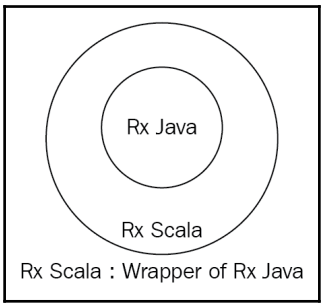
Weather Forecasting: User Systems

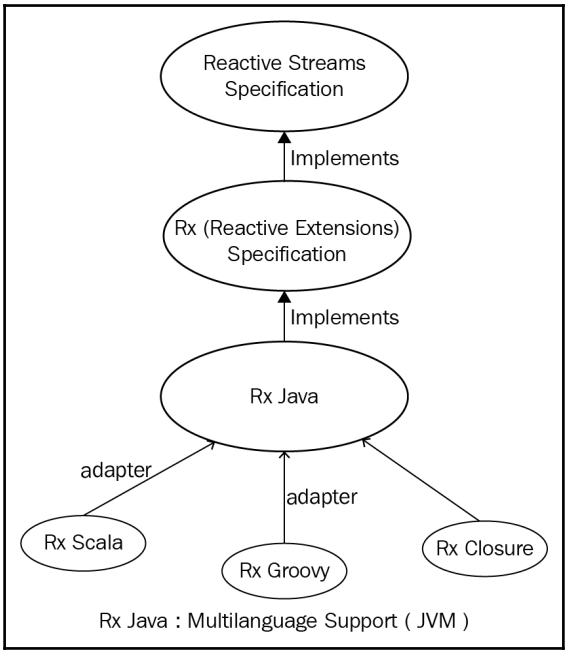
Chapter 5: Adding Reactiveness with RxScala

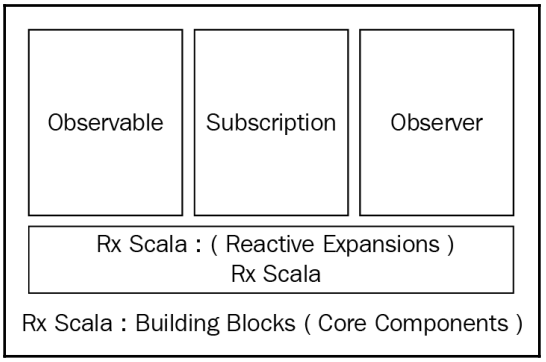


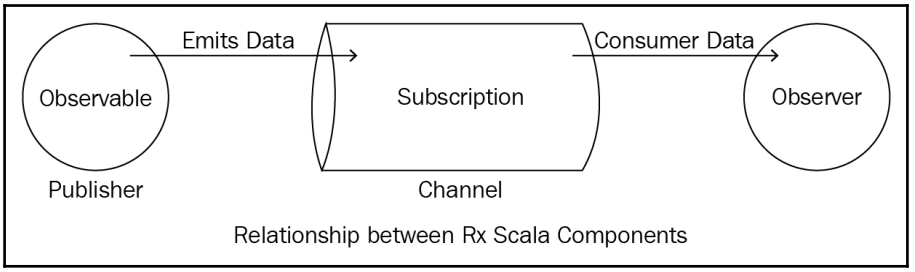


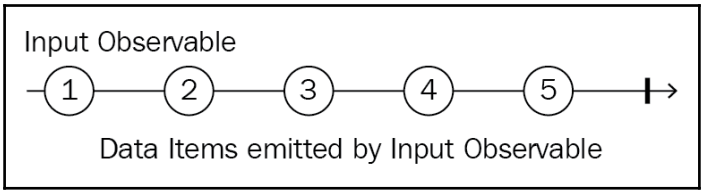

Reactive Extensions
Rx Standard
Implements
Rx Scala
Adapter
RP
RP
Scala
Rx Java
Rx Scala : Adapter on Rx Java

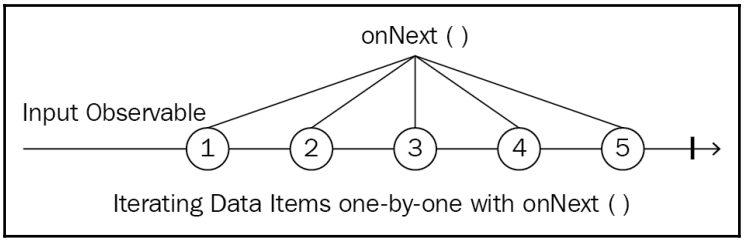

Rx Java
Rx Scala
Rx Scala : Wrapper of Rx Java

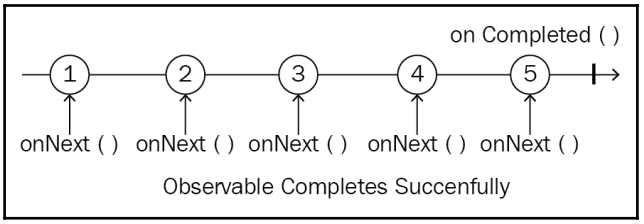

Reactive Streams
Specification
Implements
Rx (Reactive Extensions)
Specification
Implements
Rx Java
adapter
adapter
Rx Scala
Rx Groovy
Rx Closure
Rx Java : Multilanguage Support (JVM)

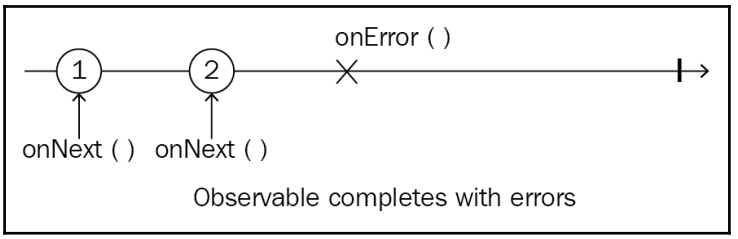

Observable
Subscription
Observer
Rx Scala : (Reactive Expansions)
Rx Scala
Rx Scala : Building Blocks (Core Components)

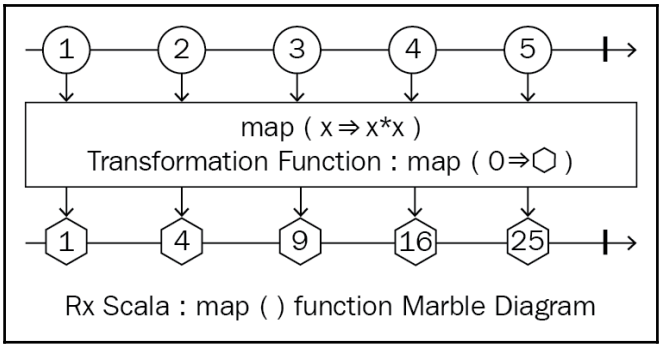

Emits Data
Consumer Data
Observable
Subscription
Observer
Publisher
Channel
Relationship between Rx Scala Components

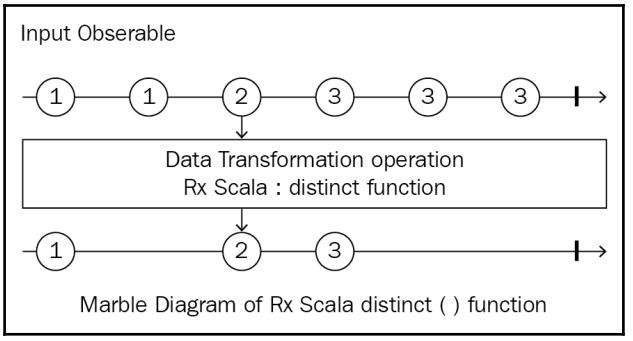

Input Observable
1
2
3
4
5
Data Items emitted by Input Observable

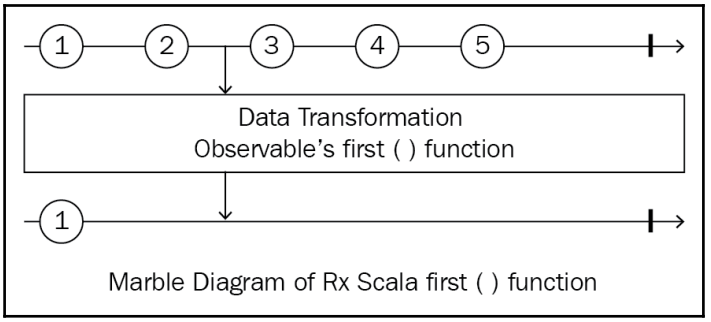

onNext ( )
Input Observable
1
2
3
4
5
Iterating Data Items one-by-one with onNext ( )


on Completed ( )
1
2
3
4
5
onNext ( )
onNext ( )
onNext ( )
onNext ( )
onNext ( )
Observable Completes Succenfully

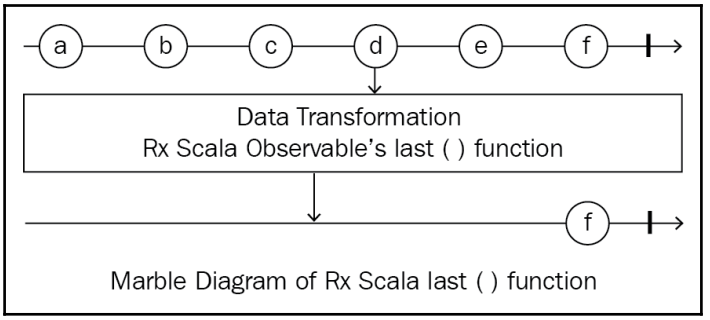

onError ( )
1
2
onNext ( )
onNext ( )
Observable completes with errors


1
2
3
4
5
map (x ⇒ x*x)
Transformation Function : map (0⇒⬡)
1
4
9
16
25
Rx Scala : map ( ) function Marble Diagram

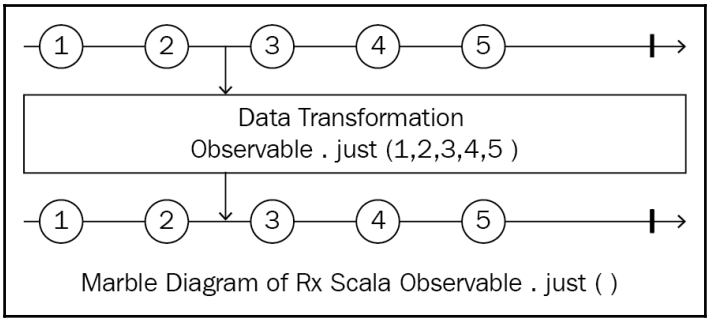

Input Obserable
1
1
2
3
3
3
Data Transformation operation
Rx Scala : distinct function
1
2
3
Marble Diagram of Rx Scala distinct ( ) function



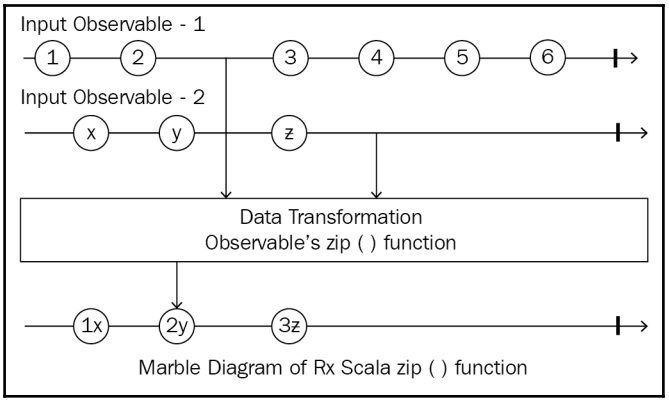
Marble Diagram of Rx Scala first ( ) function



Marble Diagram of Rx Scala last ( ) function

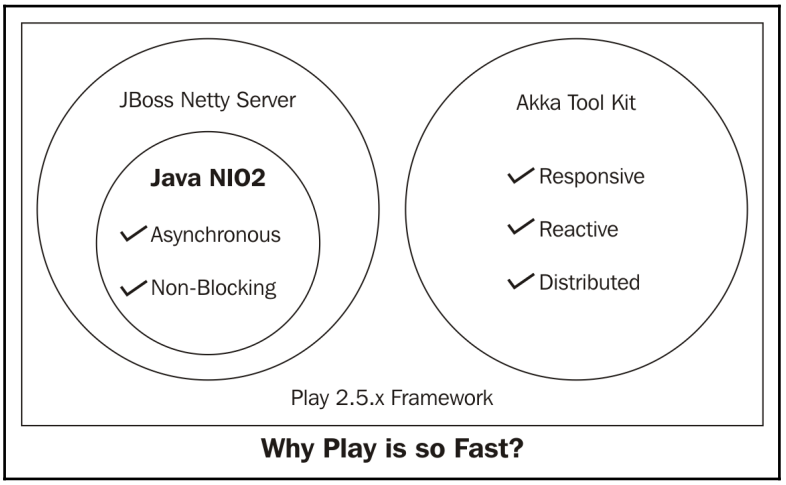


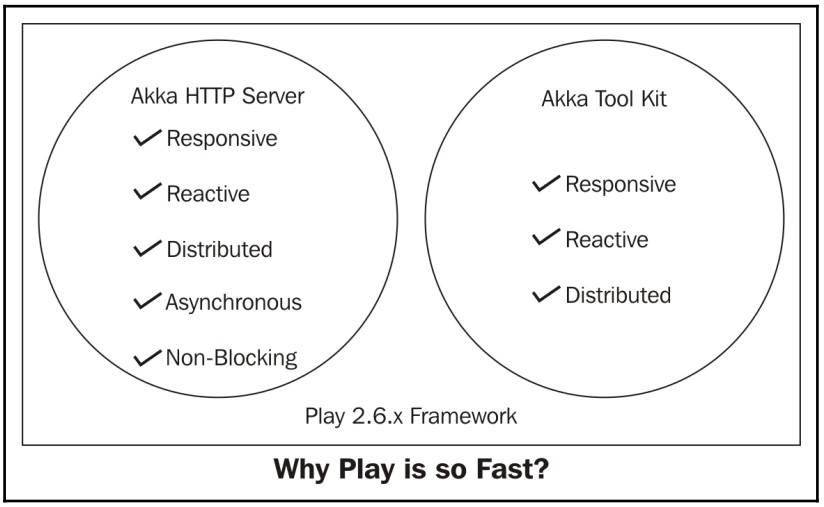
Marble Diagram of Rx Scala Observable . just ( )



Marble Diagram of Rx Scala zip ( ) function

Chapter 6: Extending Applications with Play



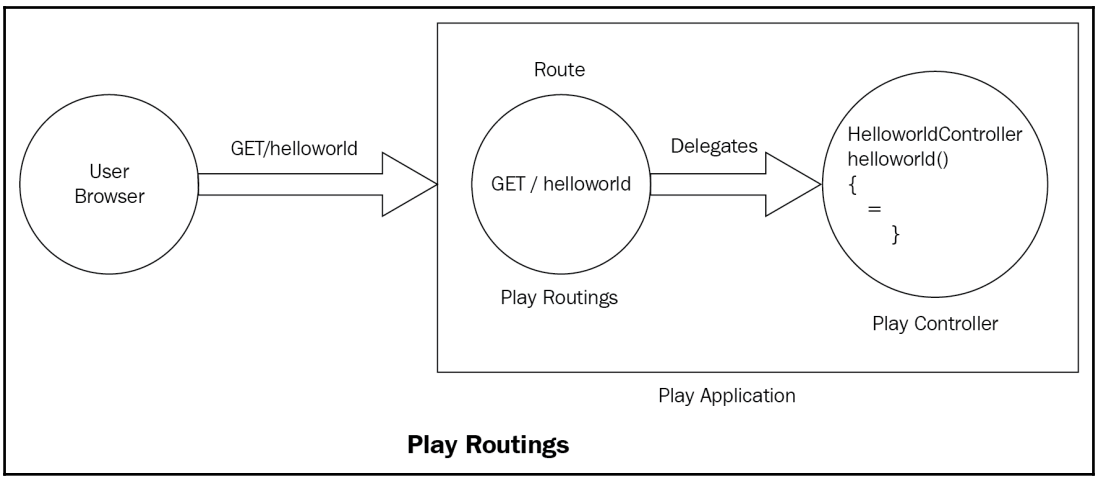


Route = <Client Request> <Controller Action>

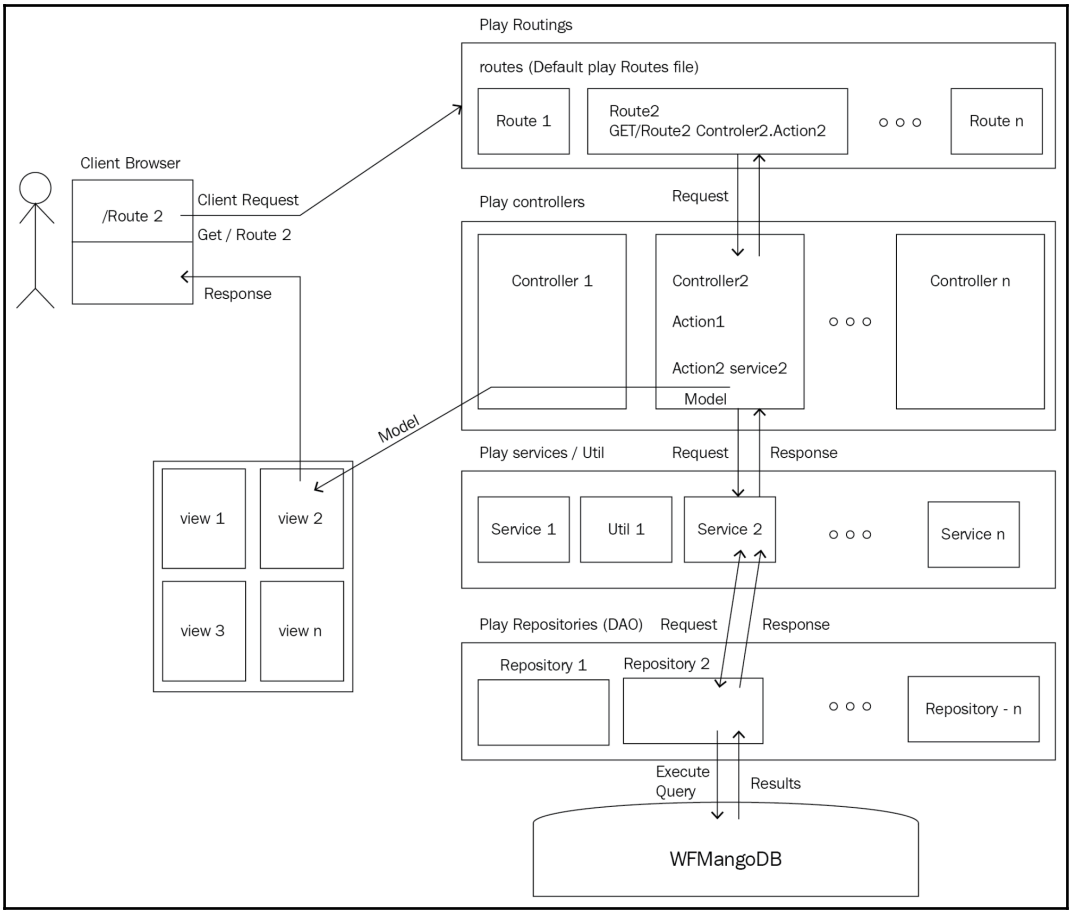
Client Request = <HTTP Request Method> <Client Request URI>

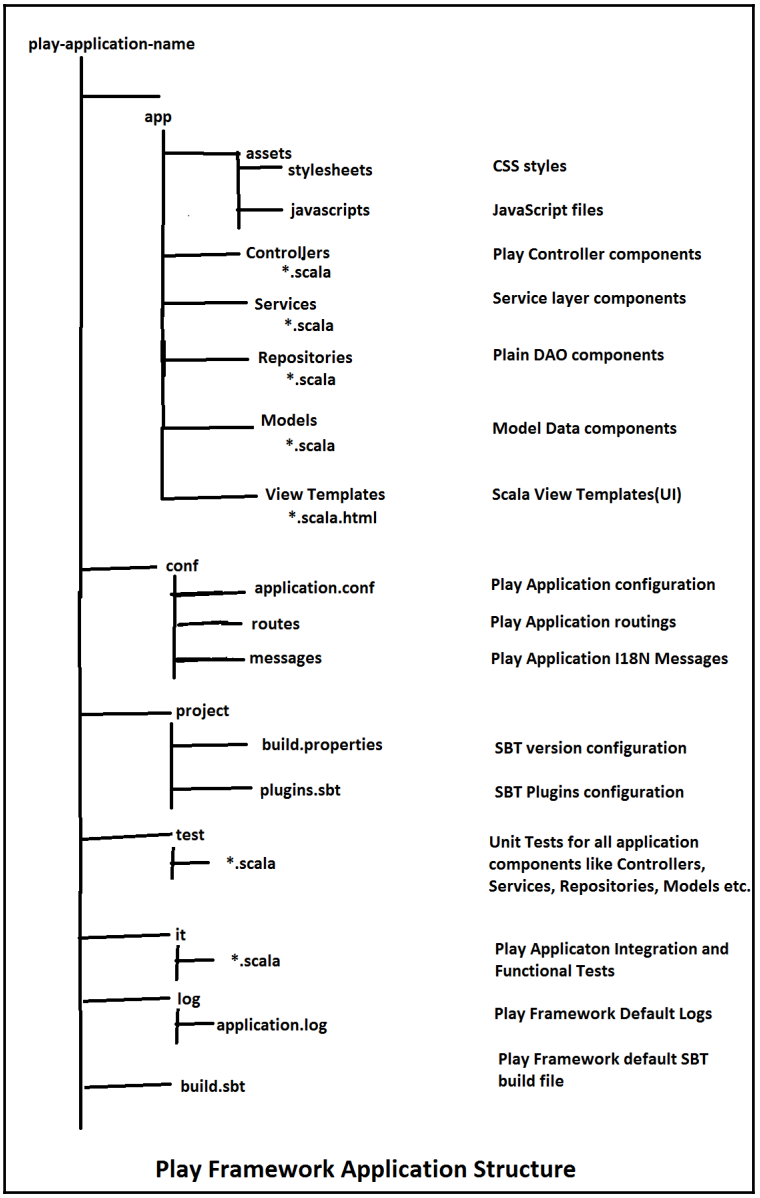
Controller Action = <Fully Qualified ControllerName>.<FunctionName>

Play Framework: Route

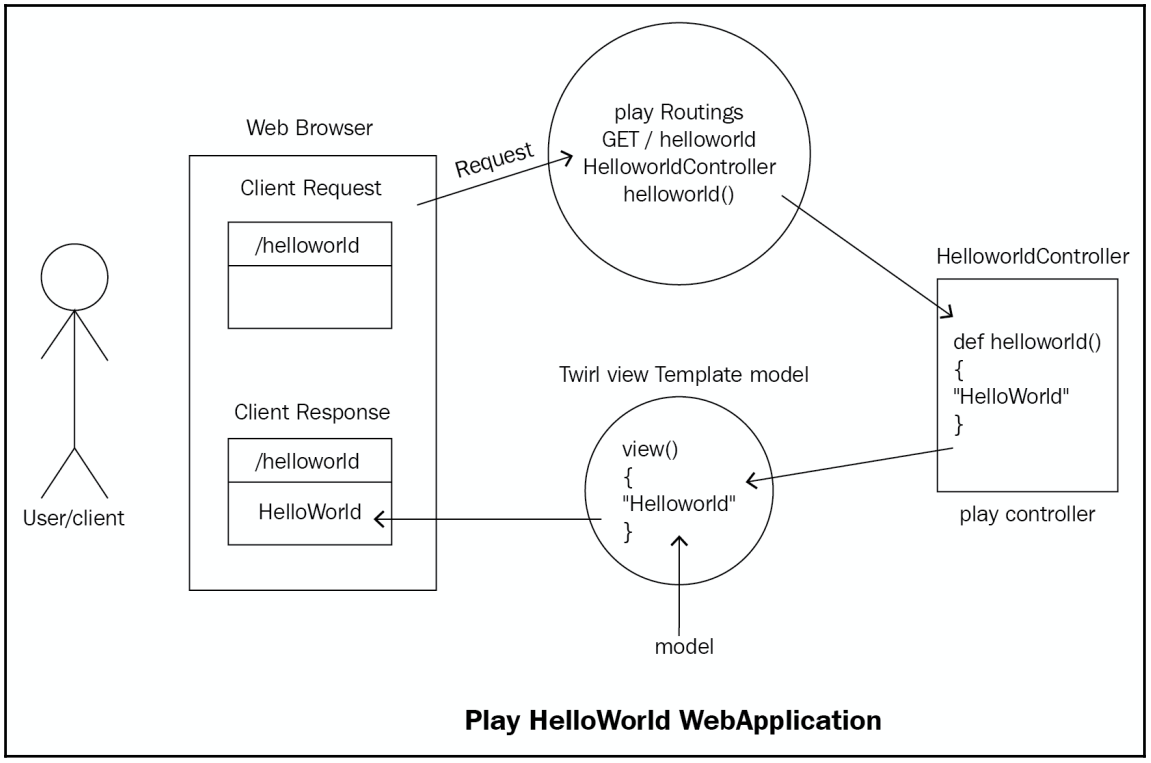


Play Routings

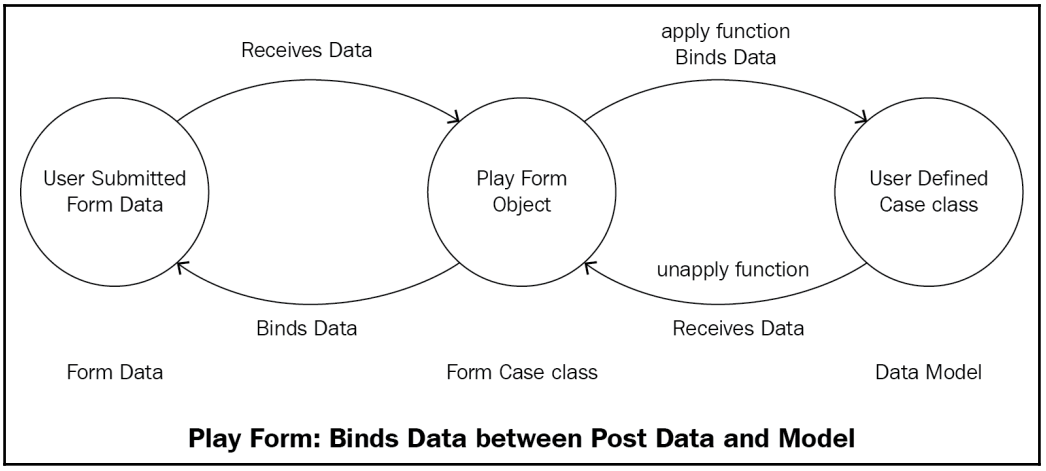

Play Routings
routes (Default play Routes file)
Route 1
Route2
GET/Route2 Controler2.Action2
Route n
Client Browser
/Route 2
Client Request
Get / Route 2
Response
Request
Play controllers
Controller 1
Controller2
Action1
Action2 service2
Model
Controller n
Model
Play services / Util
Request
Response
view 1
view 2
view 3
view n
Service 1
Util 1
Service 2
Service n
Play Repositories (DAO)
Request
Response
Repository 1
Repository 2
Repository - n
Execute
Query
Results
WFMangoDB



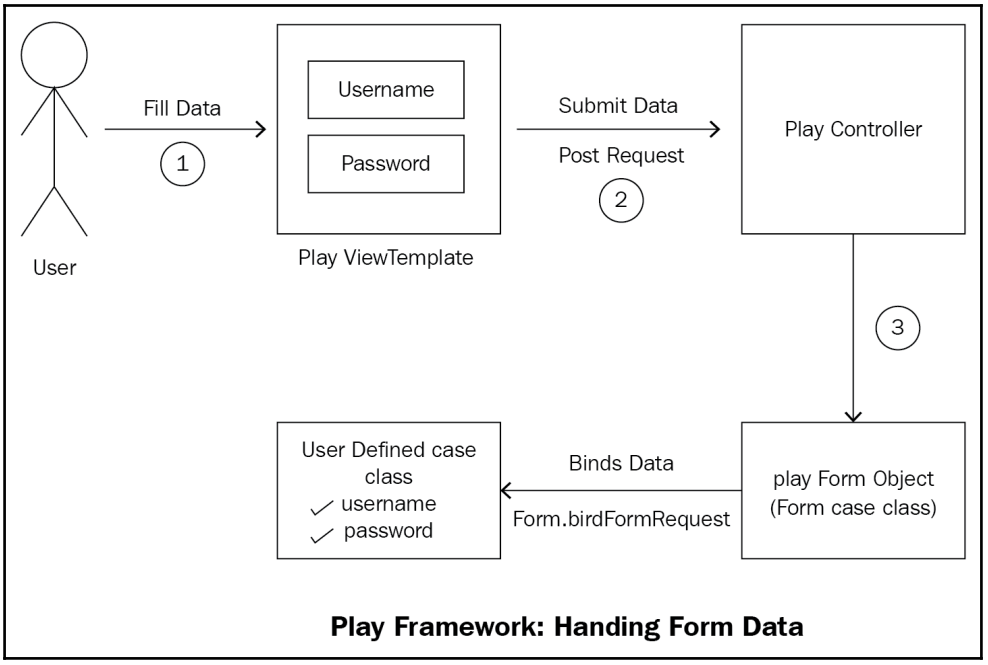
Play Framework Application Structure



Play HelloWorld WebApplication

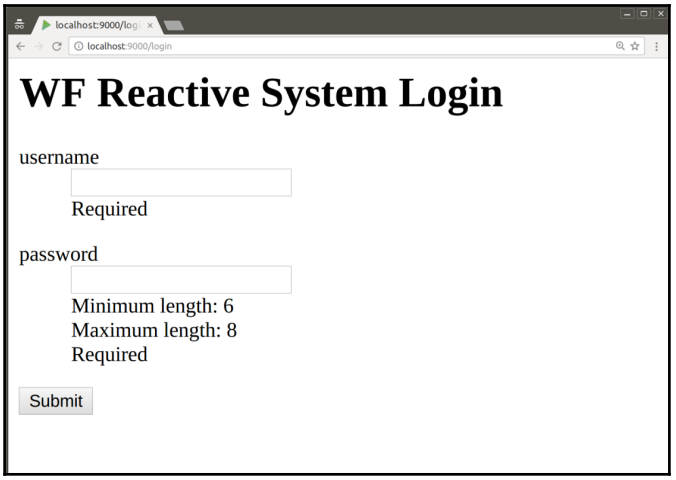


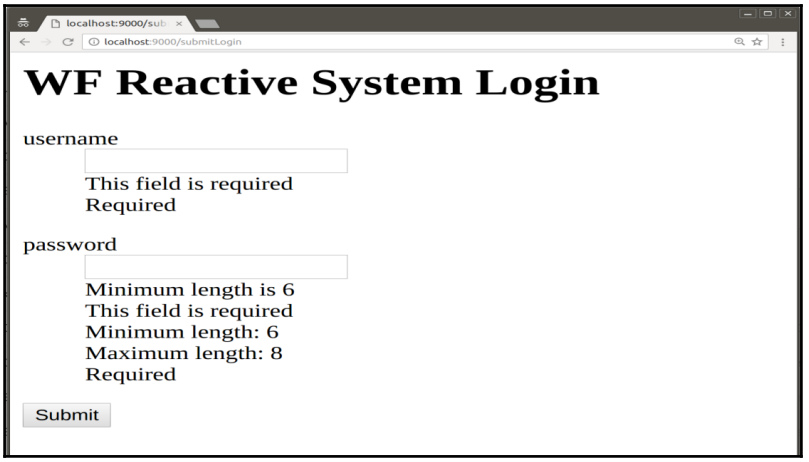
Play Form: Binds Data between Post Data and Model

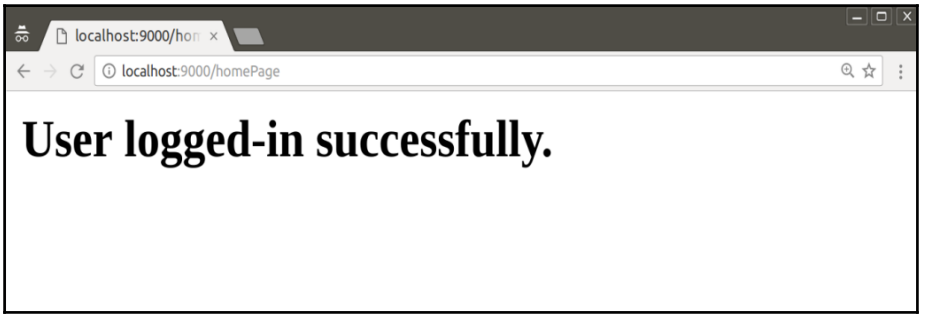


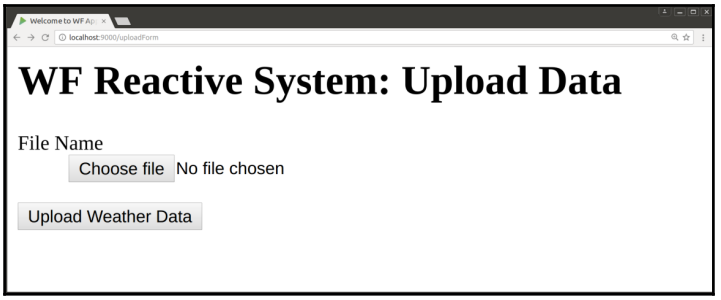
Play Framework: Handing Form Data

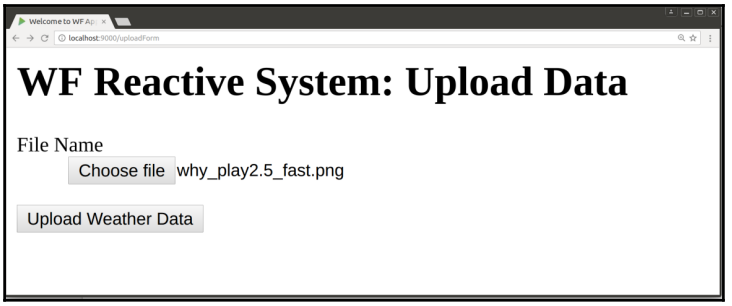
```
rambabuposa@ram: ~/Applications/RamsApps/SRPProjects/play-scala-login-form-app
rambabuposa@ram:~/Applications/RamsApps/SRPProjects/play-scala-login-form-app$ sbt run
[warn] Executing in batch mode.
[warn]   For better performance, hit [ENTER] to switch to interactive mode, or
[warn]   consider launching sbt without any commands, or explicitly passing 'shell'
[info] Loading global plugins from /home/rambabuposa/.sbt/0.13/plugins
[info] Loading project definition from /home/rambabuposa/Applications/RamsApps/SRPProjects/play-scala-login-form-app/project
[info] Set current project to play-scala-login-form-app (in build file:/home/rambabuposa/Applications/RamsApps/SRPProjects/play-scala-login-fo
rm-app/)

--- (Running the application, auto-reloading is enabled) ---

[info] p.c.s.AkkaHttpServer - Listening for HTTP on /0:0:0:0:0:0:0:0:9000

(Server started, use Enter to stop and go back to the console...)
```

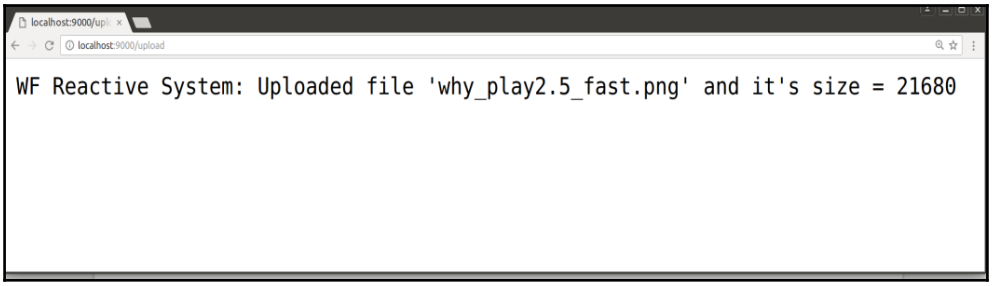

localhost:9000/login
WF Reactive System Login
username
Required
password
Minimum length: 6
Maximum length: 8
Required
Submit


localhost:9000/submitLogin
WF Reactive System Login
username
This field is required
Required
password
Minimum length is 6
This field is required
Minimum length: 6
Maximum length: 8
Required
Submit

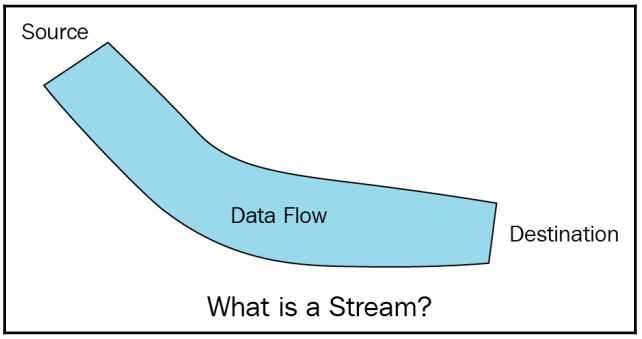

localhost:9000/homePage
User logged-in successfully.


Welcome to WF Ap
localhost:9000/uploadForm
WF Reactive System: Upload Data
File Name
Choose file
No file chosen
Upload Weather Data

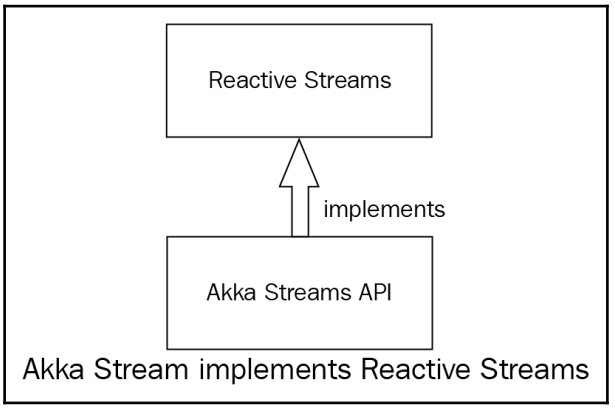

Welcome to WF Ap
localhost:9000/uploadForm
WF Reactive System: Upload Data
File Name
Choose file
why_play2.5_fast.png
Upload Weather Data


localhost:9000/up
localhost:9000/upload
WF Reactive System: Uploaded file 'why_play2.5_fast.png' and it's size = 21680

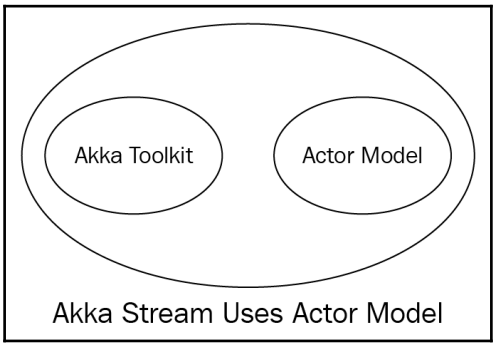
Chapter 7: Working with Reactive Streams



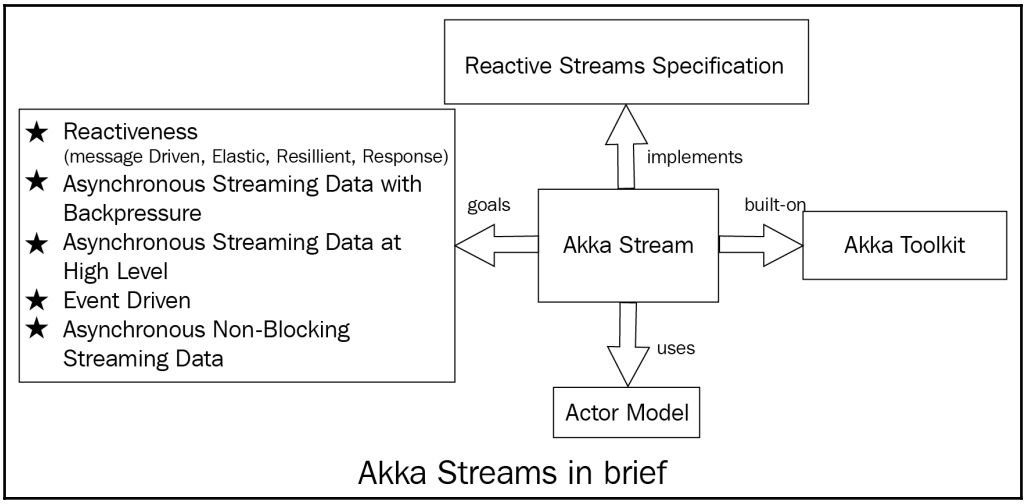
What is a Stream?

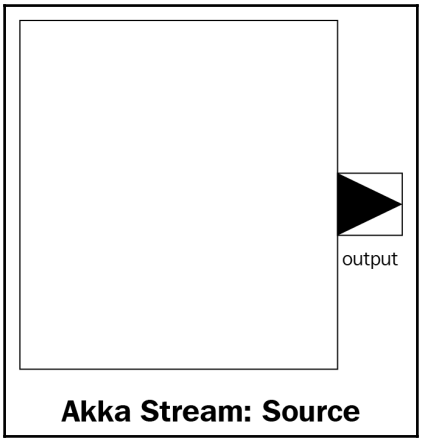


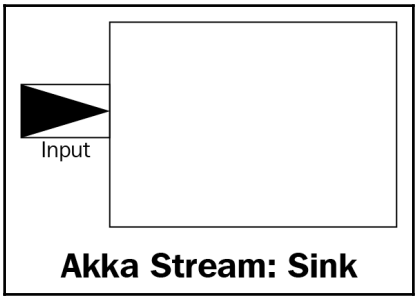
Akka Stream implements Reactive Streams

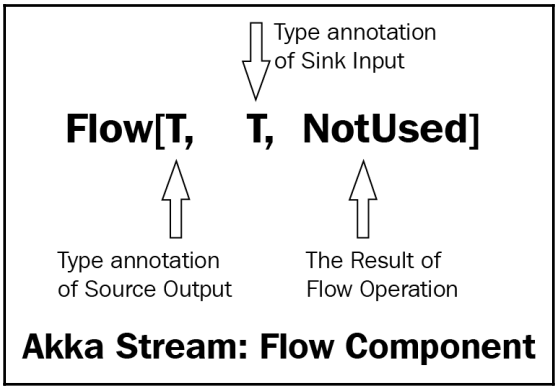


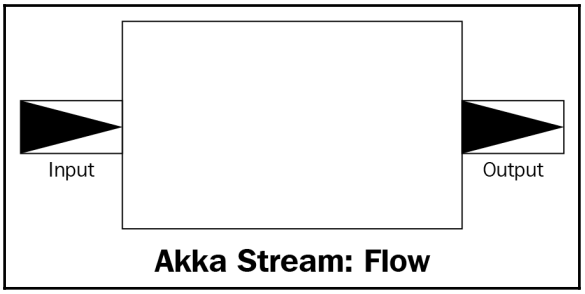
Akka Stream Uses Actor Model

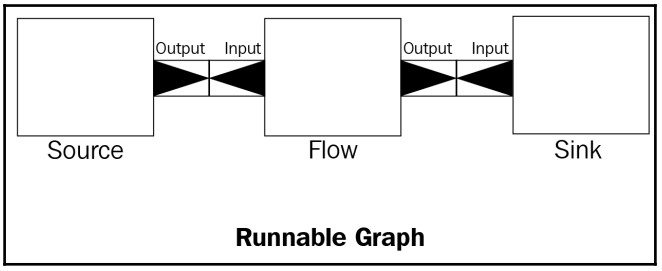

Reactive Streams Specification
★ Reactiveness
(message Driven, Elastic, Resillient, Response)
★ Asynchronous Streaming Data with Backpressure
★ Asynchronous Streaming Data at High Level
★ Event Driven
★ Asynchronous Non-Blocking Streaming Data
implements
goals
Akka Stream
built-on
Akka Toolkit
uses
Actor Model
Akka Streams in brief

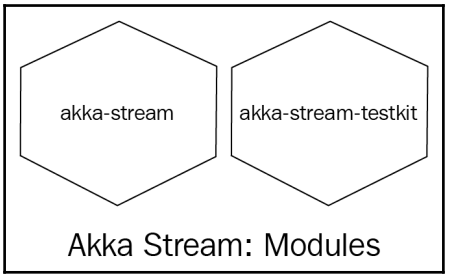

output
Akka Stream: Source


Input
Akka Stream: Sink

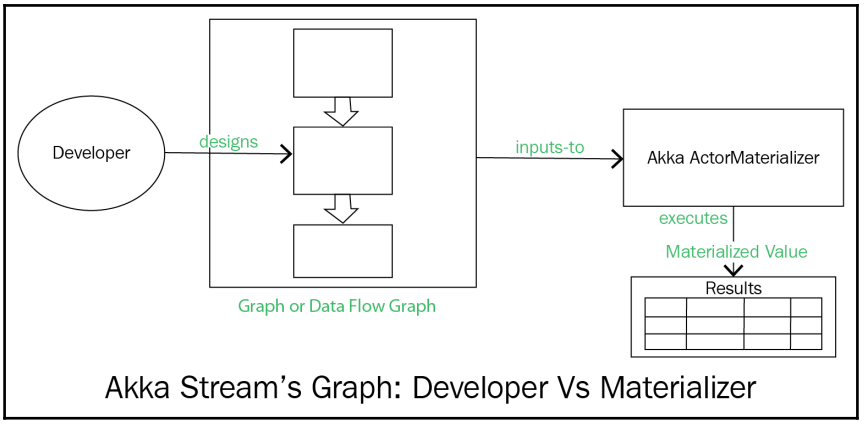

Type annotation
of Sink Input
Flow[T, T, NotUsed]
Type annotation
of Source Output
The Result of
Flow Operation
Akka Stream: Flow Component


Input
Output
Akka Stream: Flow


Output
Input
Output
Input
Source
Flow
Sink
Runnable Graph


akka-stream
akka-stream-testkit
Akka Stream: Modules

Materialize = Execute or Run



Akka Stream's Graph: Developer Vs Materializer

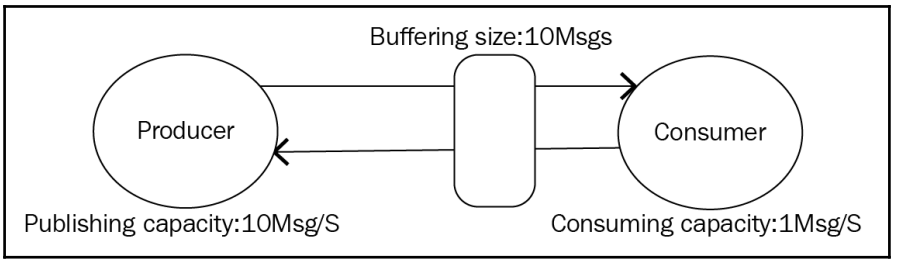
Akka Actors => Design + Execute

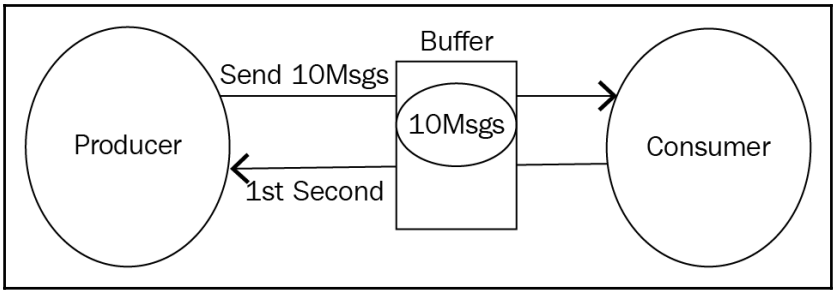
- We should create Actors
- Lots of Boilerplate
- We should write logic to process different stages
- We should pass one stage results to next

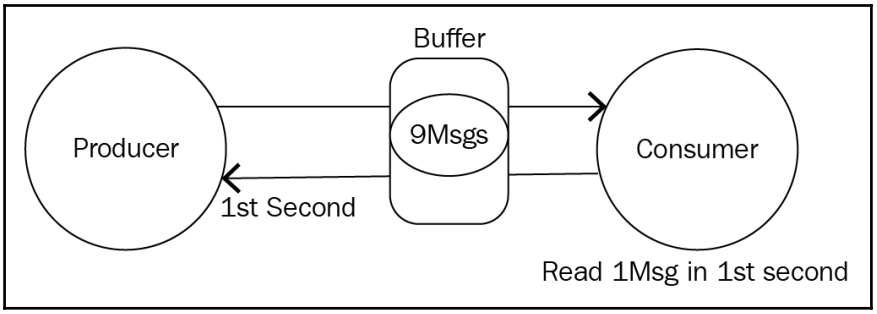
Akka Streams => Design

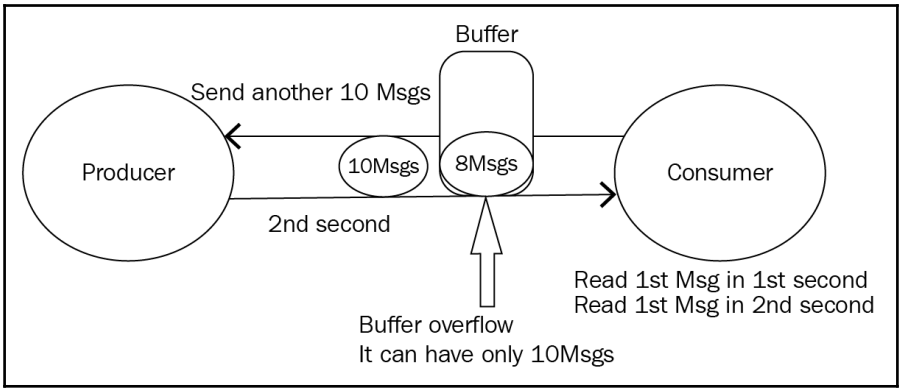
- No need to create Actors
- No Boilerplate code
- Actor Materializer will take care of executing different stages, passing them to next stage

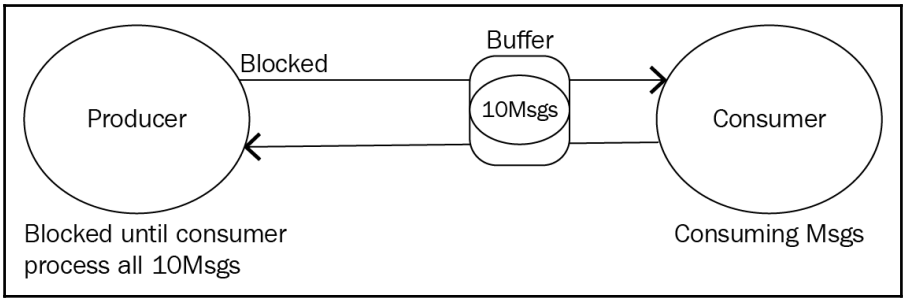
Akka Data Streaming: Actor Vs ActorMaterializer

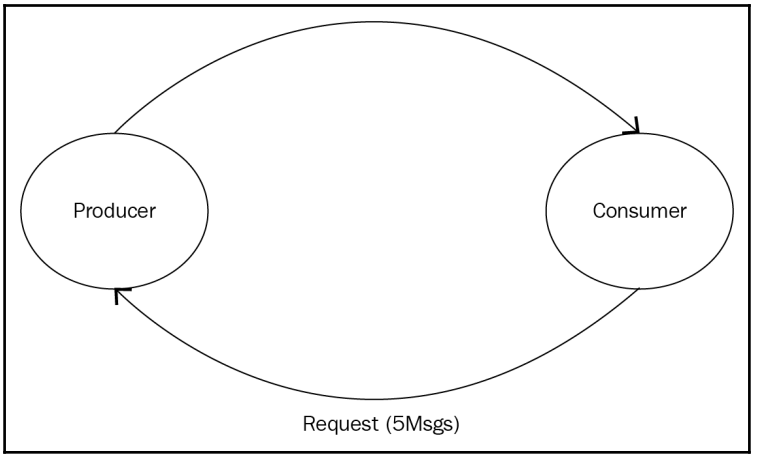


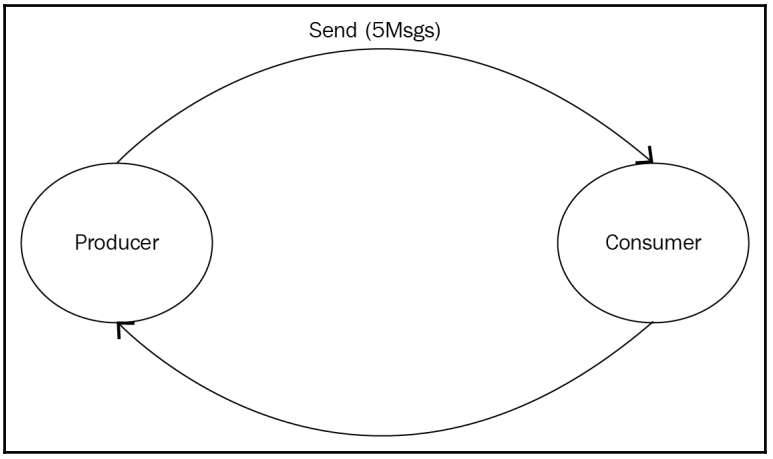

Buffer
Send 10Msgs
Producer
10Msgs
Consumer
1st Second

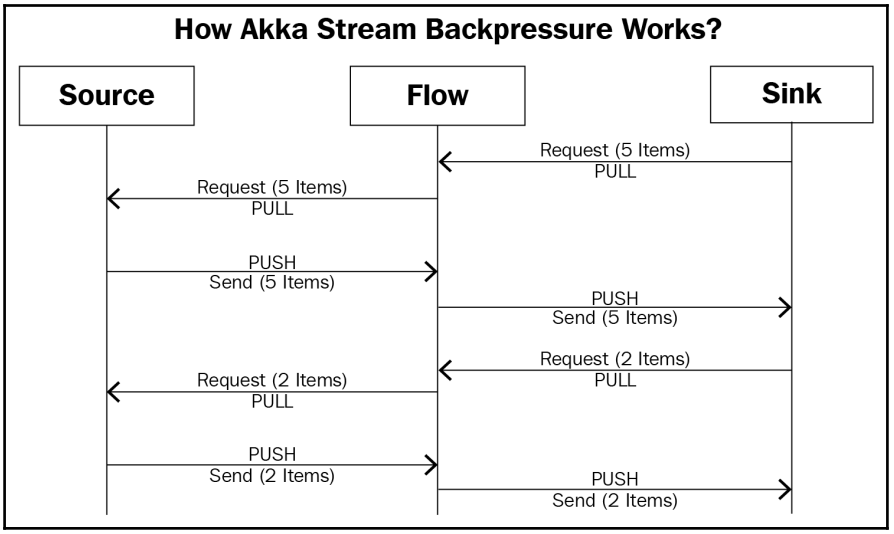

Buffer
Producer
9Msgs
Consumer
1st Second
Read 1Msg in 1st second

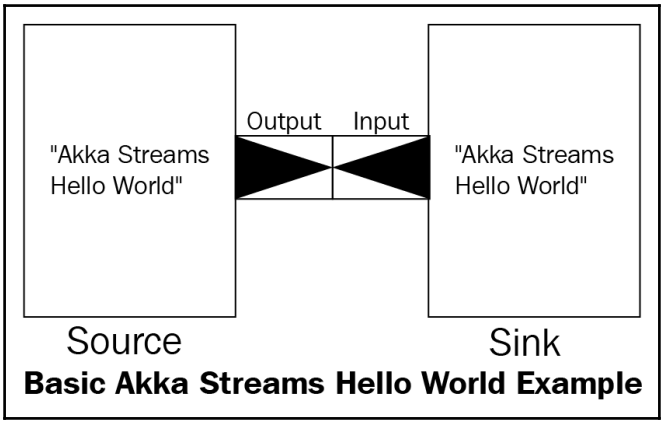

Buffer
Send another 10 Msgs
Producer
10Msgs
8Msgs
Consumer
2nd second
Read 1st Msg in 1st second
Read 1st Msg in 2nd second
Buffer overflow
It can have only 10Msgs

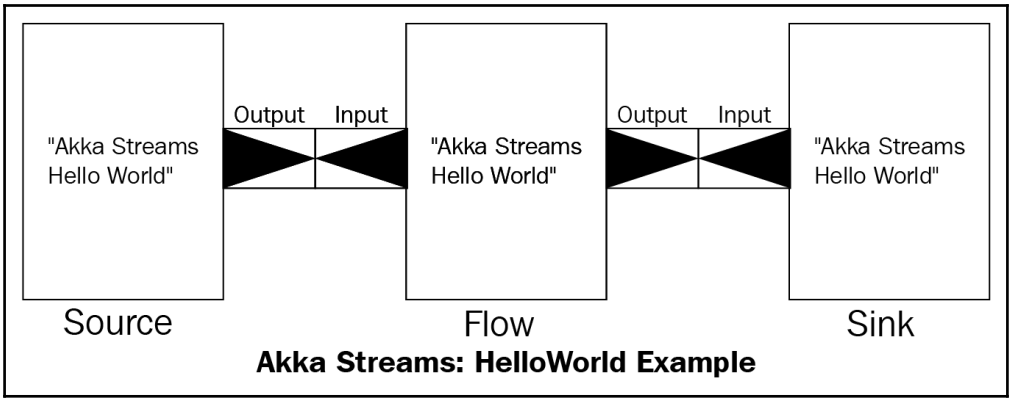

Buffer
Blocked
Producer
10Msgs
Consumer
Blocked until consumer
process all 10Msgs
Consuming Msgs


Producer
Consumer
Request (5Msgs)

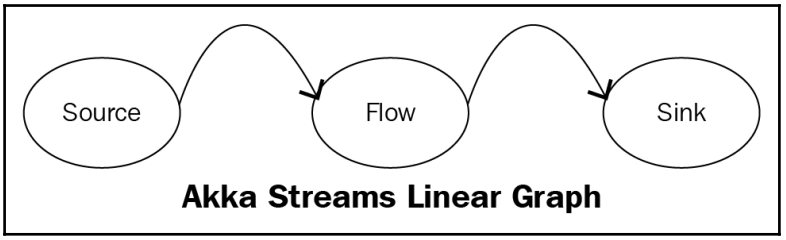

Send (5Msgs)
Producer
Consumer


How Akka Stream Backpressure Works?
Source
Flow
Sink
Request (5 Items)
PULL
Request (5 Items)
PULL
PUSH
Send (5 Items)
PUSH
Send (5 Items)
Request (2 Items)
PULL
Request (2 Items)
PULL
PUSH
Send (2 Items)
PUSH
Send (2 Items)

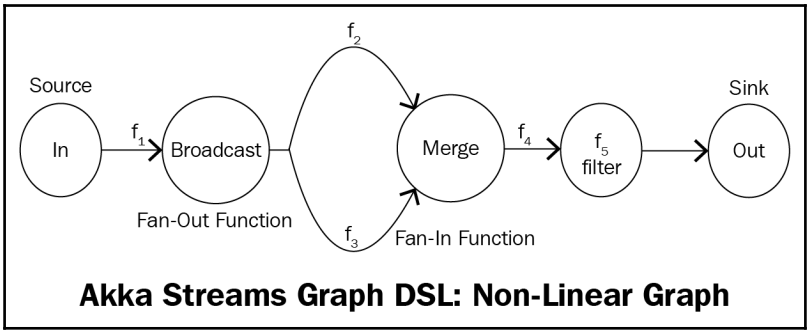

"Akka Streams Hello World"
Output
Input
"Akka Streams Hello World"
Source
Sink
Basic Akka Streams Hello World Example



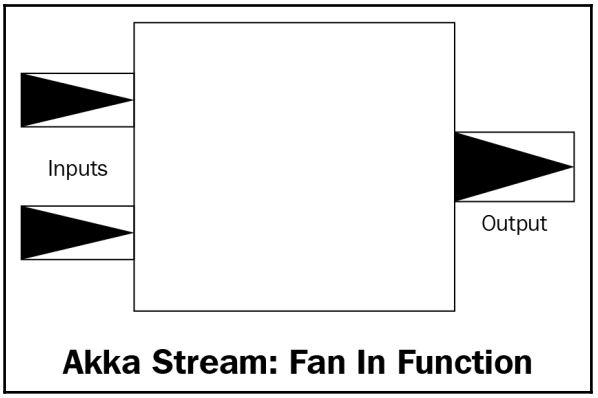
Akka Streams: HelloWorld Example

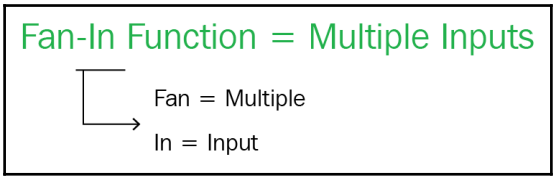


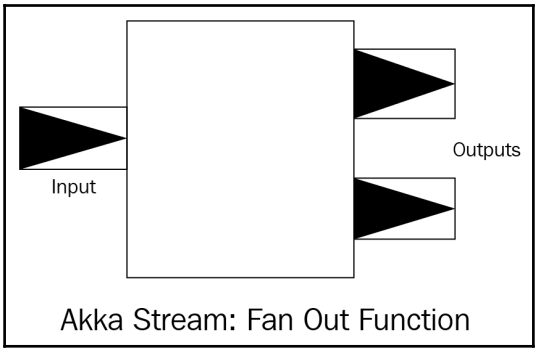
Akka Streams Linear Graph

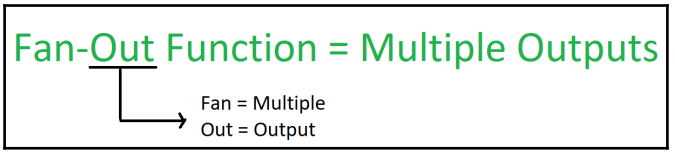


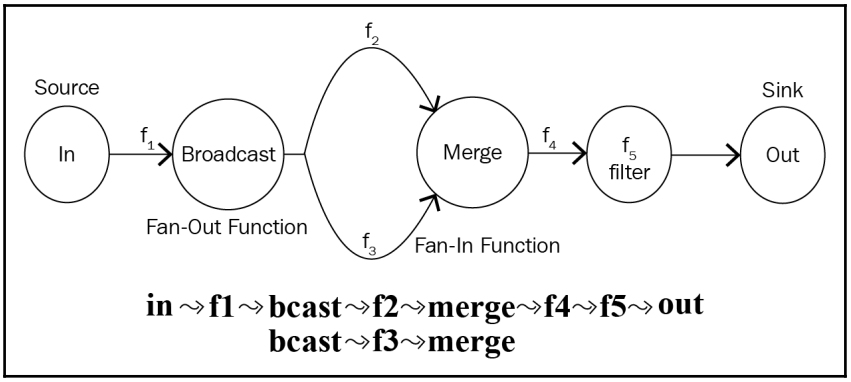
Akka Streams Graph DSL: Non-Linear Graph

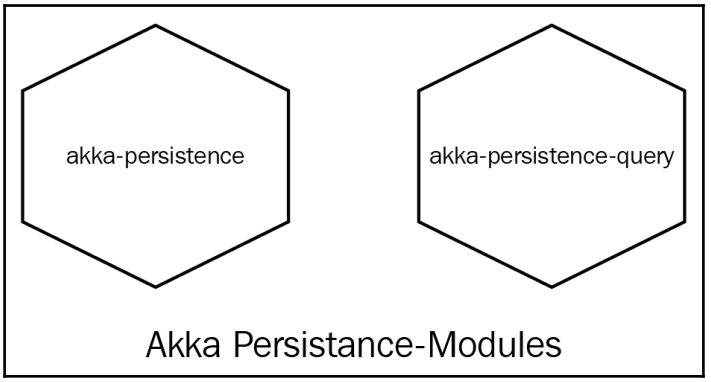

Inputs
Output
Akka Stream: Fan In Function

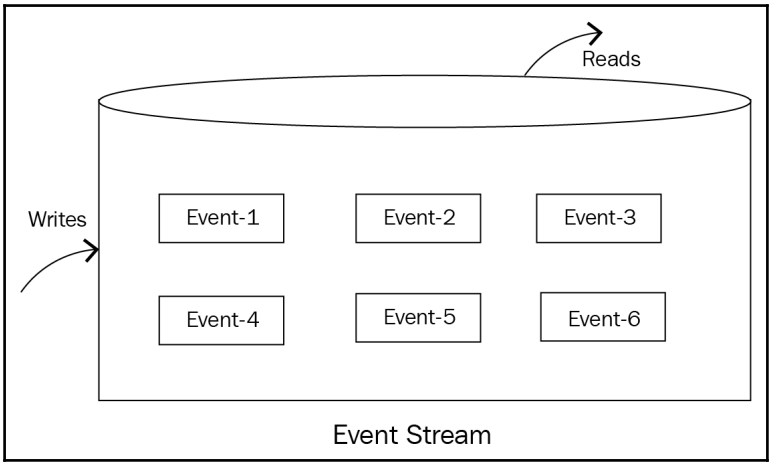

Fan-In Function = Multiple Inputs
Fan = Multiple
In = Input

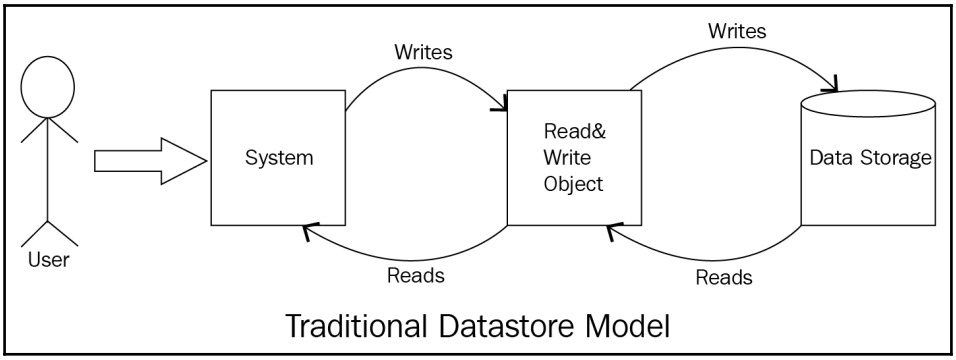

Input
Outputs
Akka Stream: Fan Out Function


Fan-Out Function = Multiple Outputs
Fan = Multiple
Out = Output

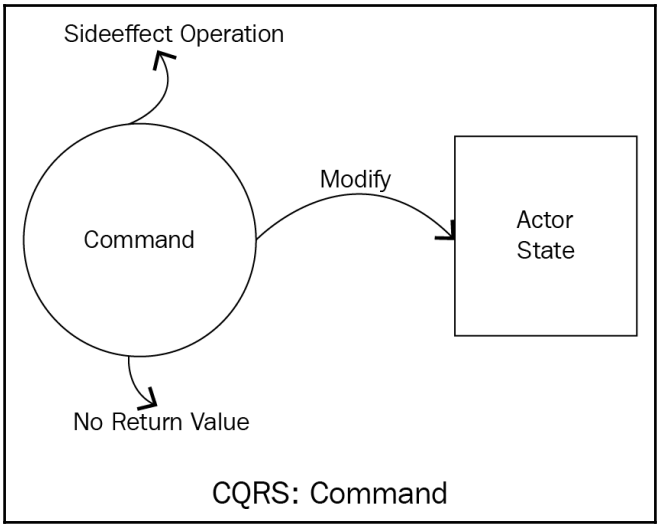

Source
In
f1
Broadcast
Fan-Out Function
f2
f3
Merge
Fan-In Function
f4
f5
filter
Sink
Out
in ⇝ f1 ⇝ bcast ⇝ f2 ⇝ merge ⇝ f4 ⇝ f5 ⇝ out
bcast ⇝ f3 ⇝ merge


akka-persistence
akka-persistence-query
Akka Persistance-Modules

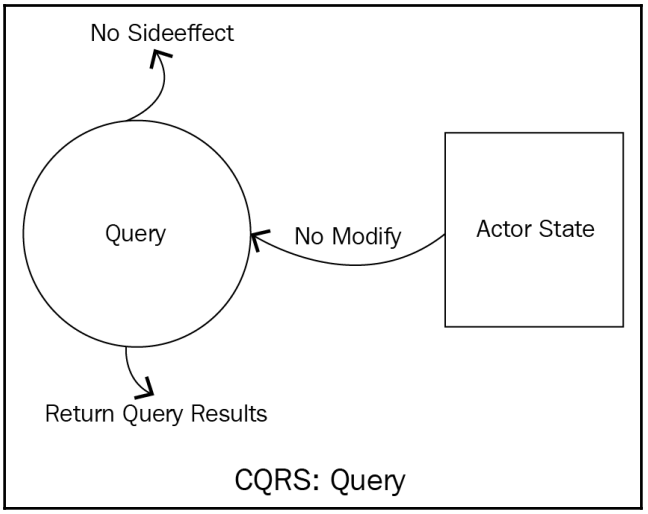

Reads
Writes
Event-1
Event-2
Event-3
Event-4
Event-5
Event-6
Event Stream

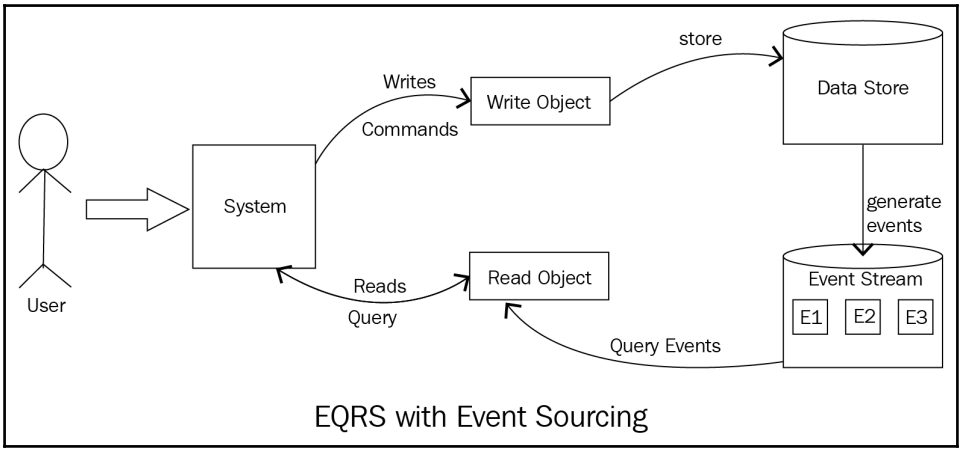


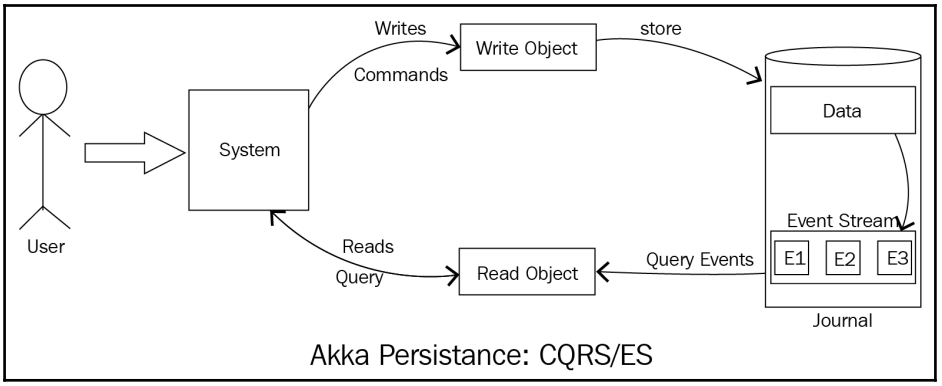
Traditional Datastore Model

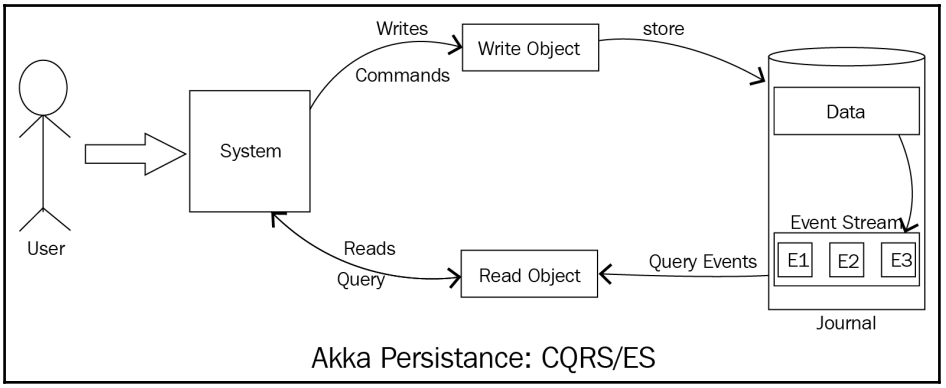


CQRS: Command

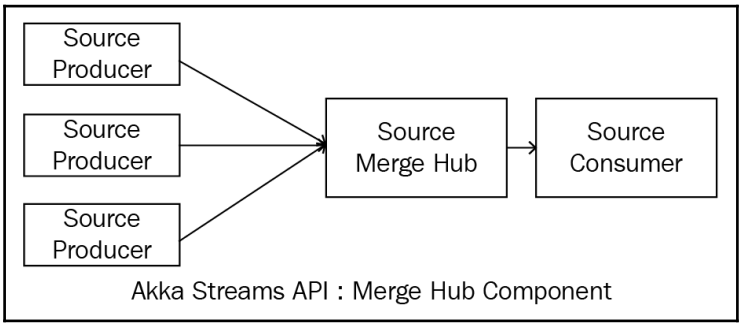

No Sideeffect
Query
No Modify
Actor State
Return Query Results
CQRS: Query


store
Writes
Write Object
Data Store
Commands
System
generate
events
Reads
Read Object
Event Stream
E1
E2
E3
User
Query
Query Events
EQRS with Event Sourcing

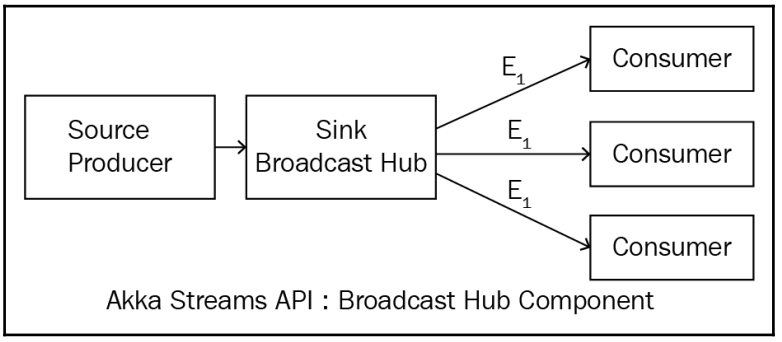

Writes
Commands
Write Object
store
Data
System
User
Event Stream
E1
E2
E3
Reads
Query
Read Object
Query Events
Journal
Akka Persistance: CQRS/ES


Writes
Commands
Write Object
store
Data
System
User
Event Stream
E1
E2
E3
Reads
Query
Read Object
Query Events
Journal
Akka Persistance: CQRS/ES

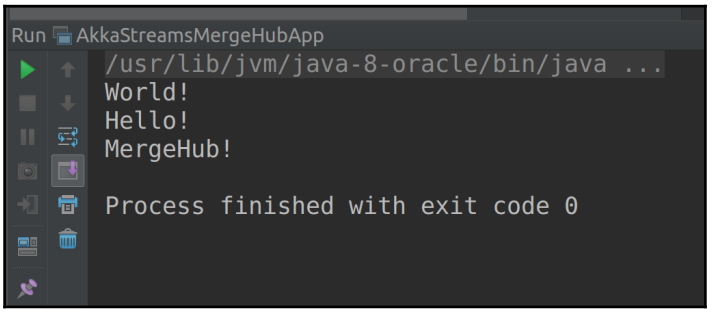
Chapter 8: Integrating Akka Streams to Play Application

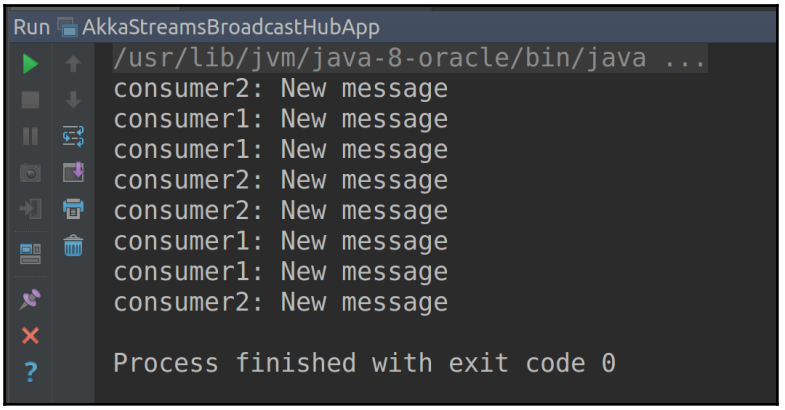


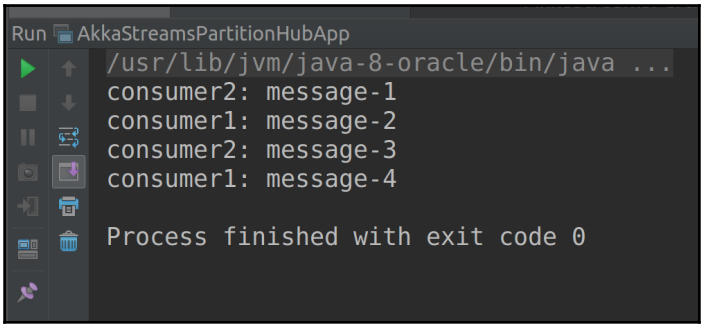
Akka Streams API : Merge Hub Component

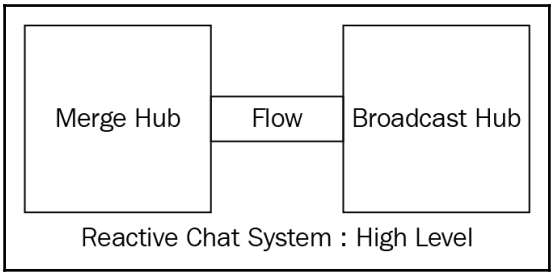


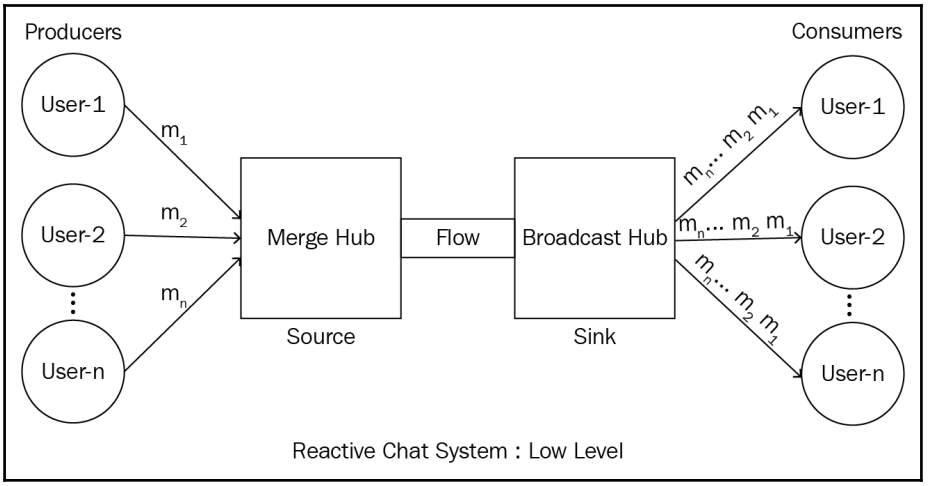
Akka Streams API : Broadcast Hub Component

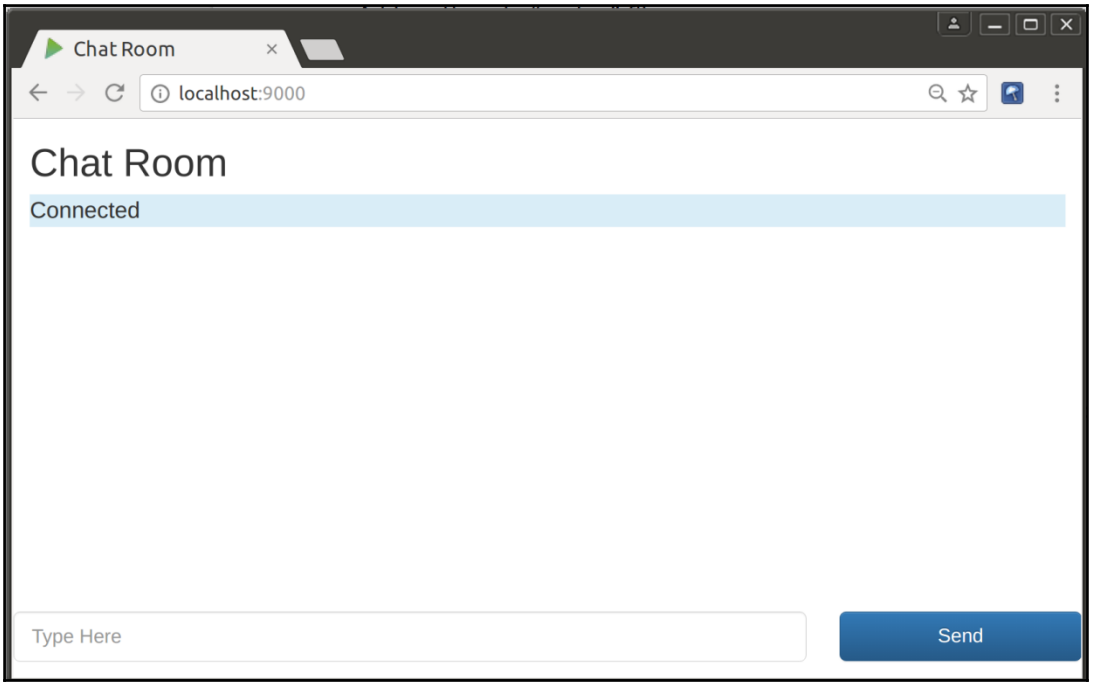


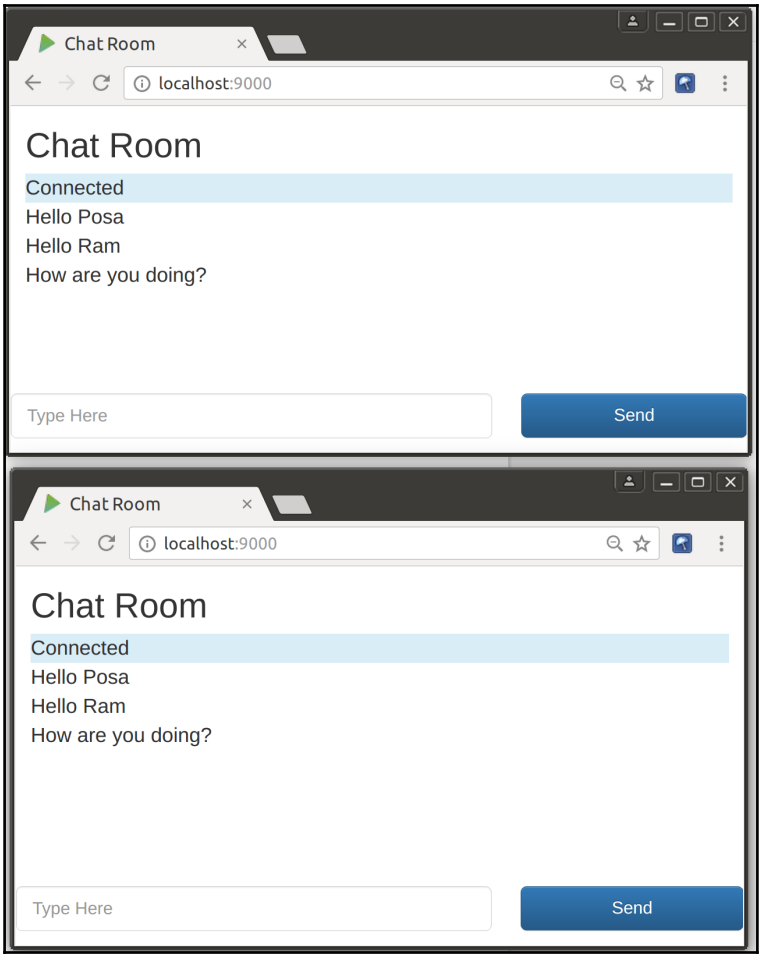

Run AkkaStreamsBroadcastHubApp
/usr/lib/jvm/java-8-oracle/bin/java ...
consumer2: New message
consumer1: New message
consumer1: New message
consumer2: New message
consumer2: New message
consumer1: New message
consumer1: New message
consumer2: New message
Process finished with exit code 0

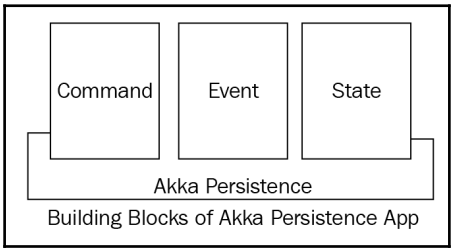

Run AkkaStreamsPartitionHubApp
/usr/lib/jvm/java-8-oracle/bin/java ...
consumer2: message-1
consumer1: message-2
consumer2: message-3
consumer1: message-4
Process finished with exit code 0


Merge Hub
Flow
Broadcast Hub
Reactive Chat System : High Level


Producers
Consumers
User-1
User-2
User-n
m1
m2
mn
Merge Hub
Source
Flow
Broadcast Hub
Sink
mn... m2 m1
User-1
User-2
User-n
Reactive Chat System : Low Level

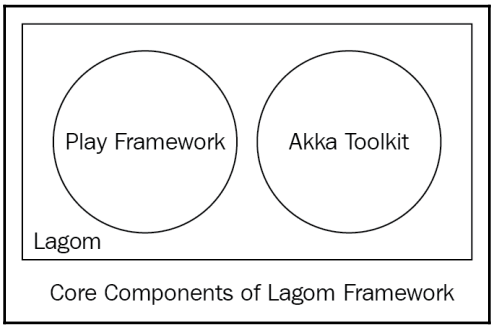

Chat Room
localhost:9000
Chat Room
Connected
Type Here
Send

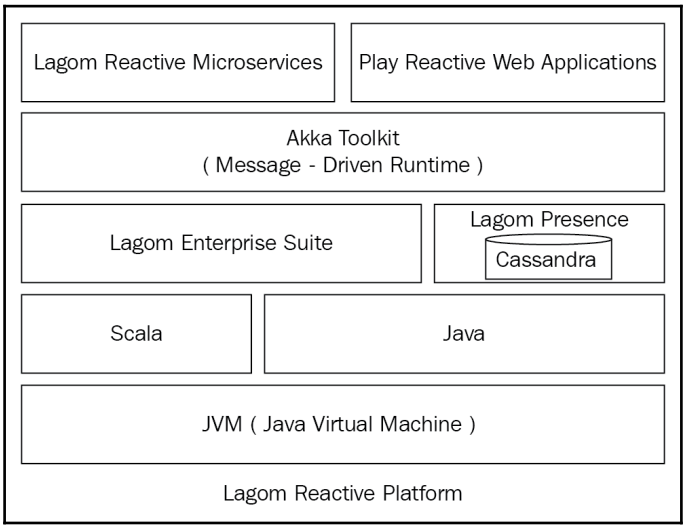


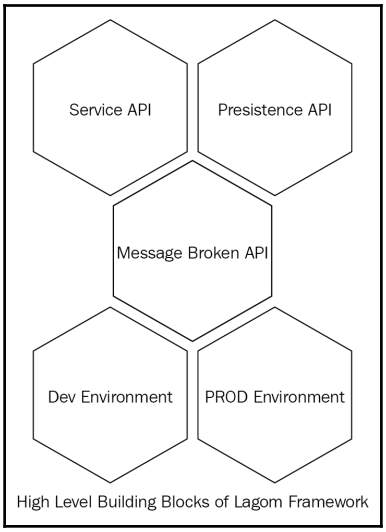


Building Blocks of Akka Persistence App

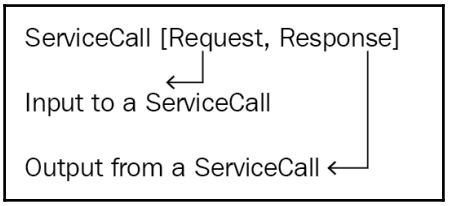
Chapter 9: Reactive Microservices with Lagom

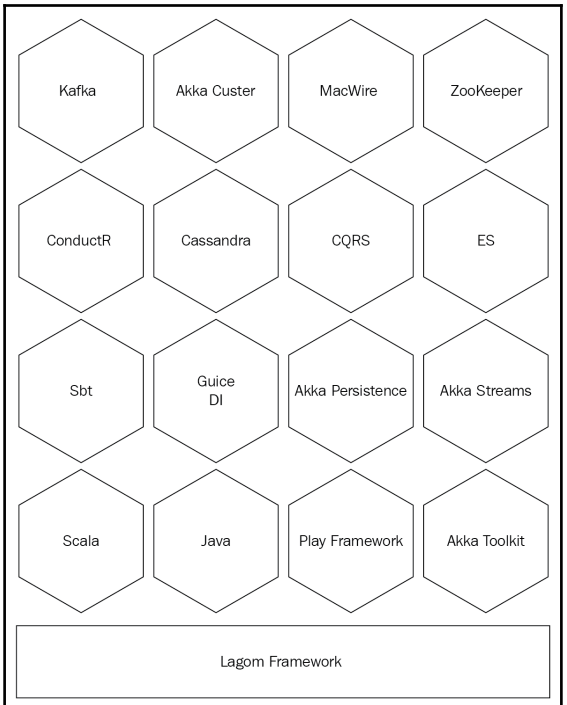






High Level Building Blocks of Lagom Framework





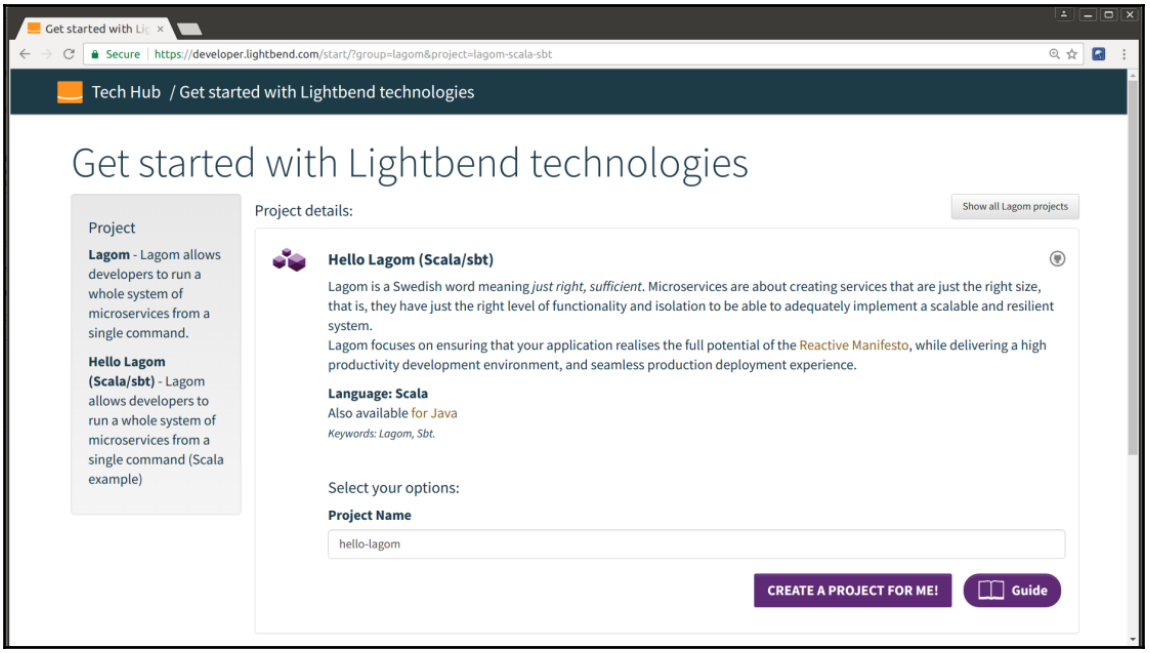
Internal components of Lagom Framework

```
Lagom - project - name
        ├- lagom - project - name - api
        ├- lagom - project - name - implementation
```

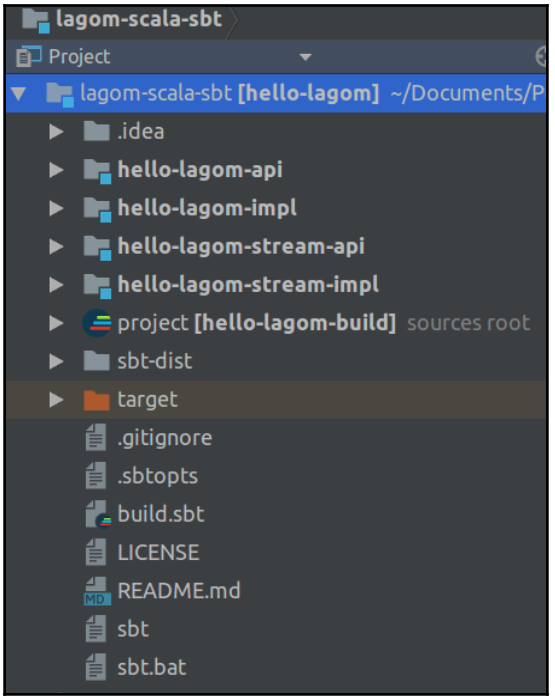
High Level Lagom Project Structure

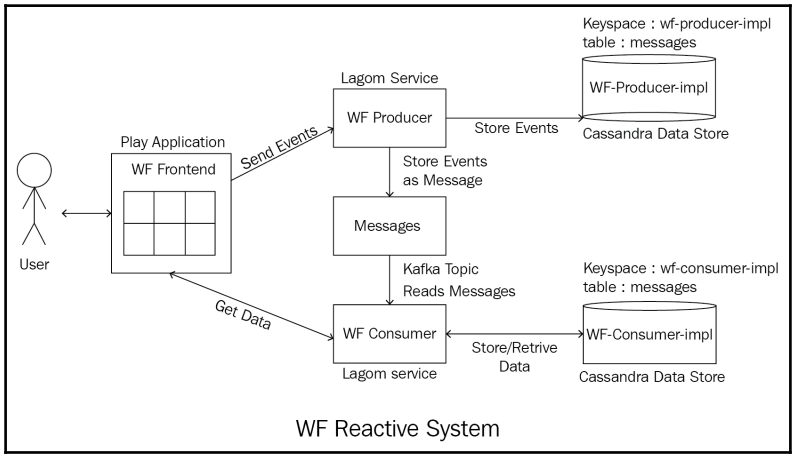
```
Lagom - reactive - system - name
        ├- lagom - servicename - api
        ├- lagom - servicename - impl
        ├- lagom - servicename - frontend
```

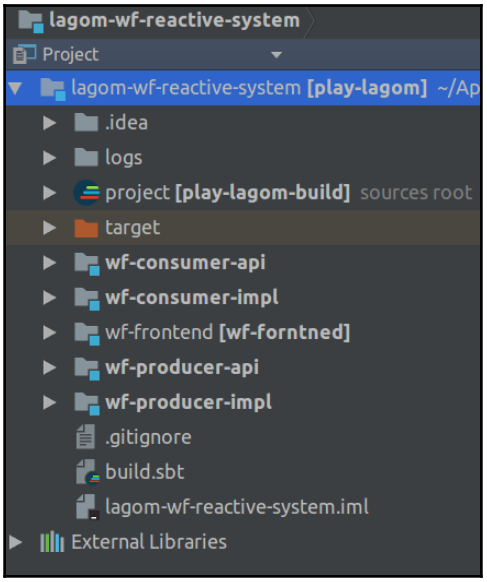
High Level Lagom Project Structure With Frontend

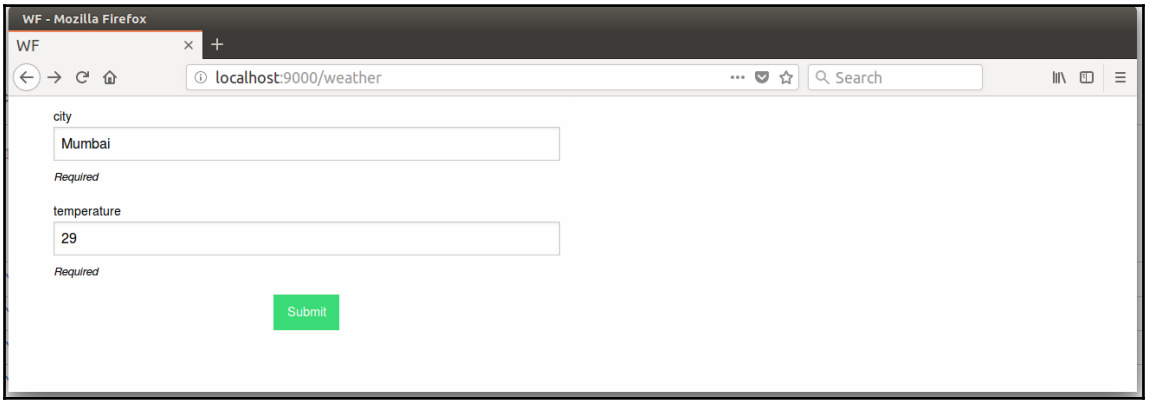

Get started with Li
Secure
https://developer.lightbend.com/start/?group=lagom&project=lagom-scala-sbt
Tech Hub / Get started with Lightbend technologies
Get started with Lightbend technologies
Project
Lagom - Lagom allows developers to run a whole system of microservices from a single command.
Hello Lagom (Scala/sbt) - Lagom allows developers to run a whole system of microservices from a single command (Scala example)
Project details:
Show all Lagom projects
Hello Lagom (Scala/sbt)
Lagom is a Swedish word meaning just right, sufficient. Microservices are about creating services that are just the right size, that is, they have just the right level of functionality and isolation to be able to adequately implement a scalable and resilient system.
Lagom focuses on ensuring that your application realises the full potential of the Reactive Manifesto, while delivering a high productivity development environment, and seamless production deployment experience.
Language: Scala
Also available for Java
Keywords: Lagom, Sbt.
Select your options:
Project Name
hello-lagom
CREATE A PROJECT FOR ME!
Guide

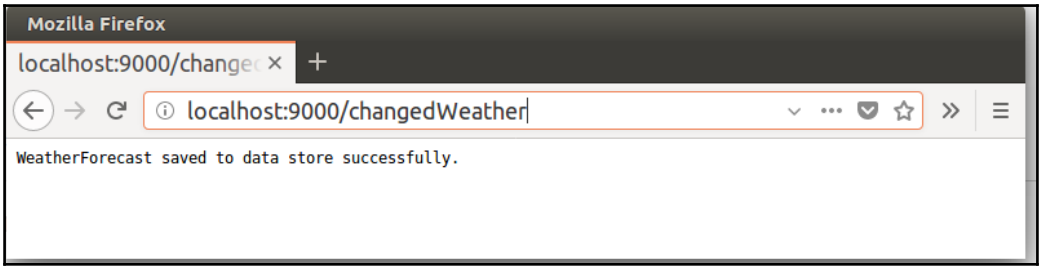

Mozilla Firefox
localhost:9000/api/he
localhost:9000/api/hello/World
190%
Hello, World!

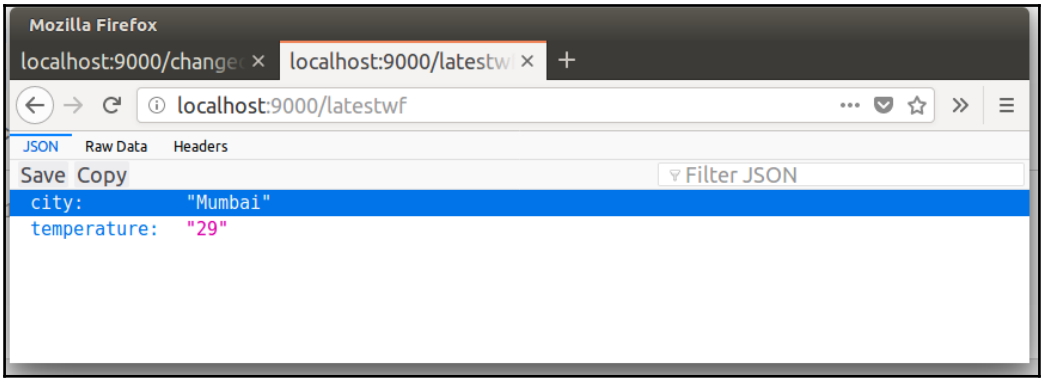

lagom-scala-sbt
Project
lagom-scala-sbt [hello-lagom] ~/Documents/P
.idea
hello-lagom-api
hello-lagom-impl
hello-lagom-stream-api
hello-lagom-stream-impl
project [hello-lagom-build] sources root
sbt-dist
target
.gitignore
.sbtopts
build.sbt
LICENSE
README.md
sbt
sbt.bat

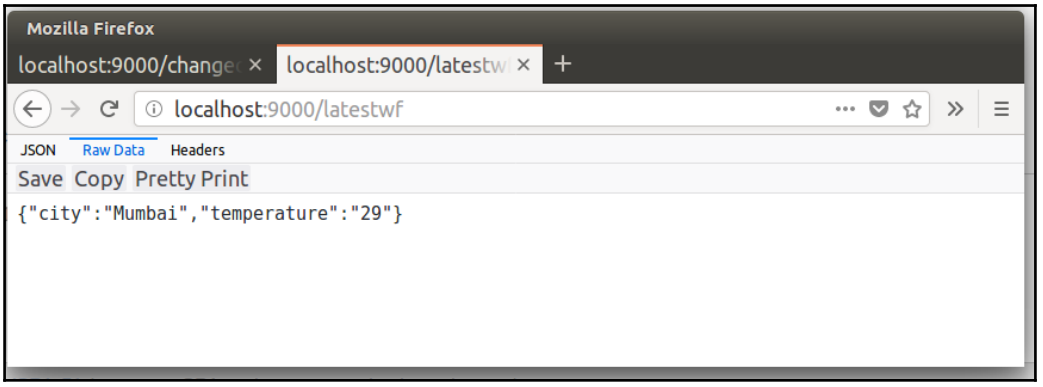

Keyspace : wf-producer-impl
table : messages
Lagom Service
WF-Producer-impl
WF Producer
Store Events
Cassandra Data Store
Play Application
Send Events
WF Frontend
Store Events
as Message
User
Messages
Kafka Topic
Reads Messages
Keyspace : wf-consumer-impl
table : messages
Get Data
WF Consumer
WF-Consumer-impl
Store/Retrive
Data
Lagom service
Cassandra Data Store
WF Reactive System

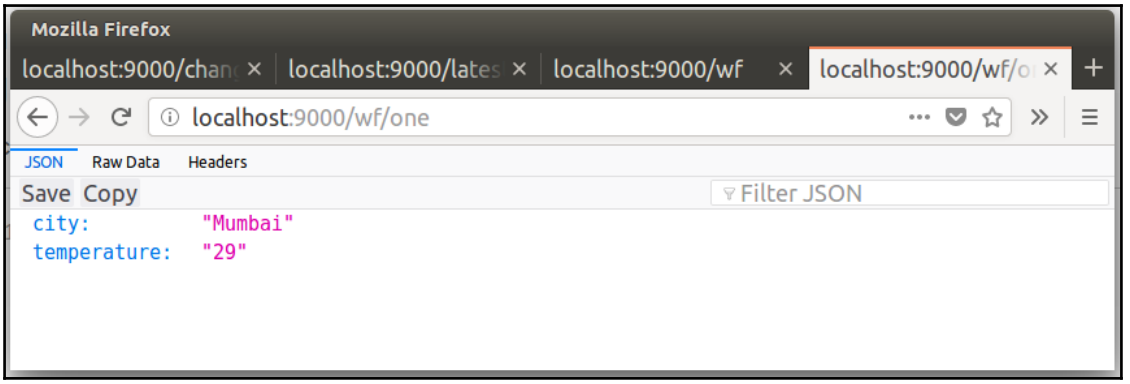

lagom-wf-reactive-system
Project
lagom-wf-reactive-system [play-lagom] ~/Ap
.idea
logs
project [play-lagom-build] sources root
target
wf-consumer-api
wf-consumer-impl
wf-frontend [wf-forntned]
wf-producer-api
wf-producer-impl
.gitignore
build.sbt
lagom-wf-reactive-system.iml
External Libraries

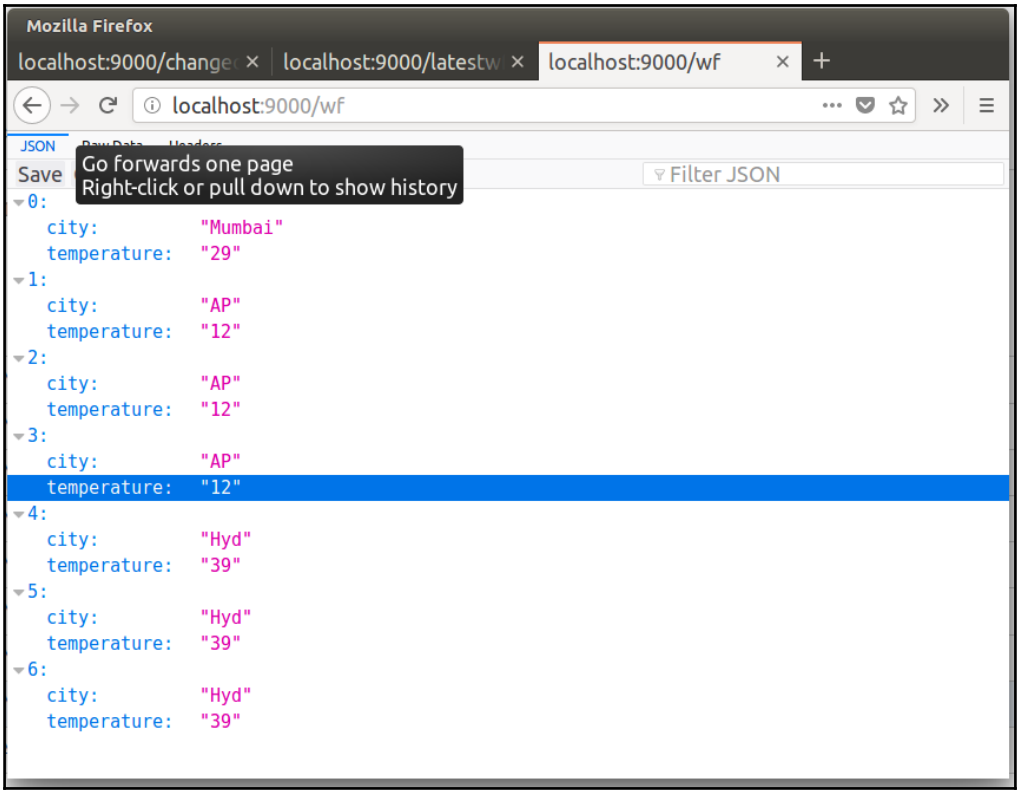

WF - Mozilla Firefox
WF
localhost:9000/weather
Search
city
Mumbai
Required
temperature
29
Required
Submit


Mozilla Firefox
localhost:9000/change
localhost:9000/changedWeather
WeatherForecast saved to data store successfully.

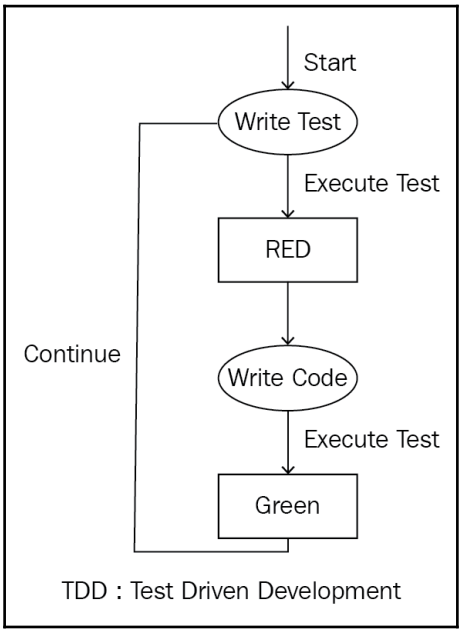

Mozilla Firefox
localhost:9000/latestwf
JSON
Raw Data
Headers
Save
Copy
Filter JSON
city: "Mumbai"
temperature: "29"


Mozilla Firefox
localhost:9000/latestwf
JSON
Raw Data
Headers
Save
Copy
Pretty Print
{"city":"Mumbai","temperature":"29"}

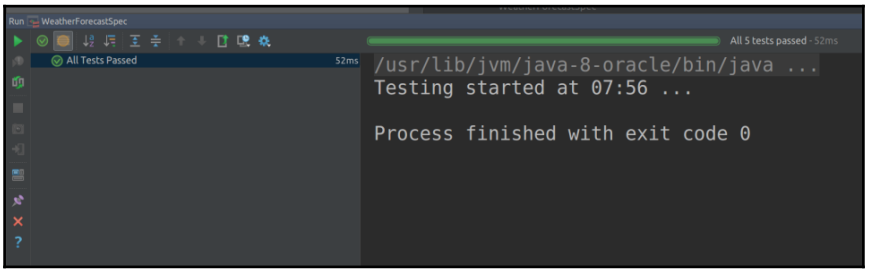

Mozilla Firefox
localhost:9000/wf/one
JSON
Raw Data
Headers
Save
Copy
Filter JSON
city: "Mumbai"
temperature: "29"


Mozilla Firefox
localhost:9000/change
localhost:9000/latestw
localhost:9000/wf
localhost:9000/wf
JSON
Save
Go forwards one page
Right-click or pull down to show history
Filter JSON
0:
city: "Mumbai"
temperature: "29"
1:
city: "AP"
temperature: "12"
2:
city: "AP"
temperature: "12"
3:
city: "AP"
temperature: "12"
4:
city: "Hyd"
temperature: "39"
5:
city: "Hyd"
temperature: "39"
6:
city: "Hyd"
temperature: "39"

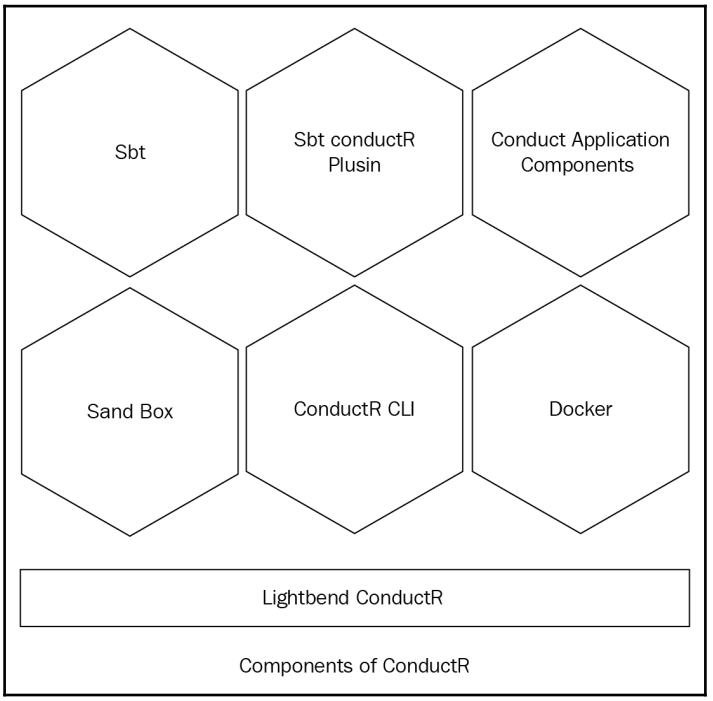
Chapter 10: Testing Reactive Microservices



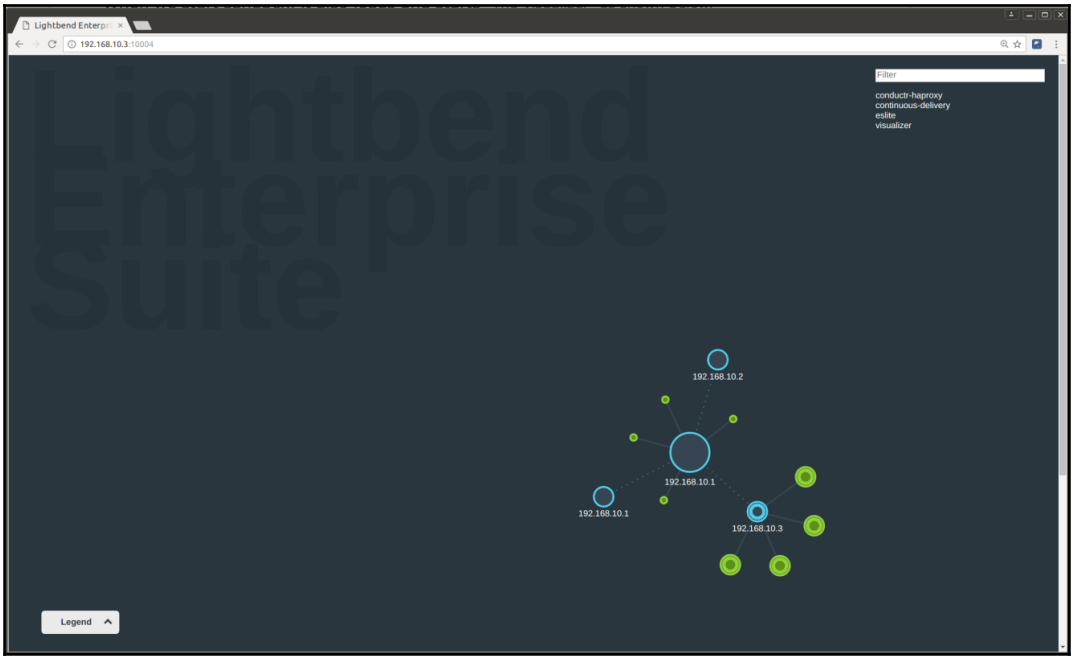
Scale Test Plus ⇒ Scale Test +Play Framework

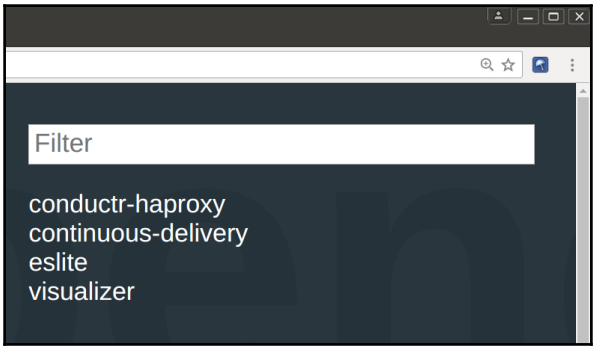


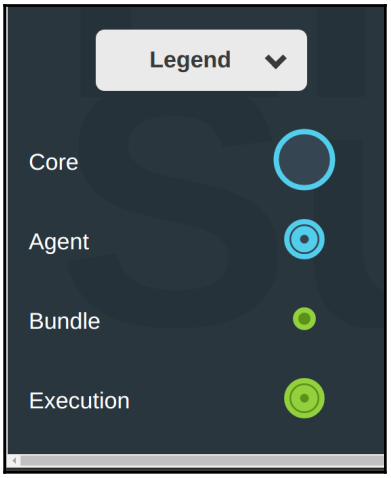
Chapter 11: Managing Microservices in ConductR



Components of ConductR


Lightbend Enterpri
192.168.10.3:10004
Filter
conductr-haproxy
continuous-delivery
eslite
visualizer
192.168.10.2
192.168.10.1
192.168.10.1
192.168.10.3
Legend

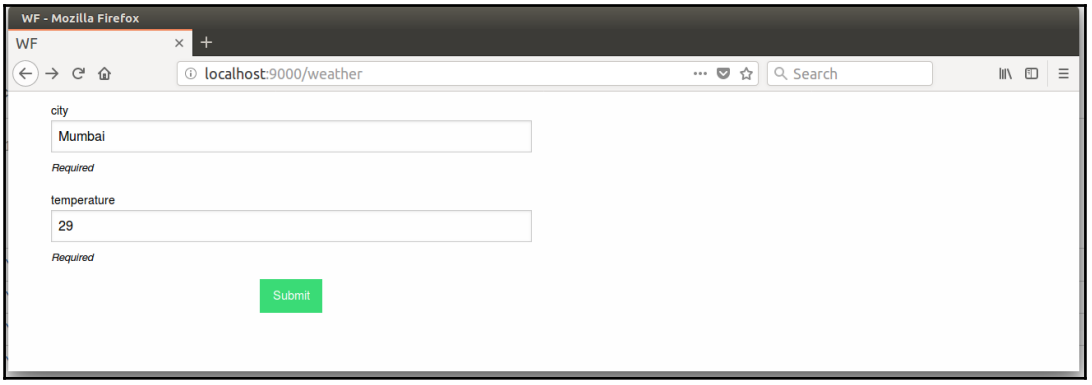

Filter
conductr-haproxy
continuous-delivery
eslite
visualizer

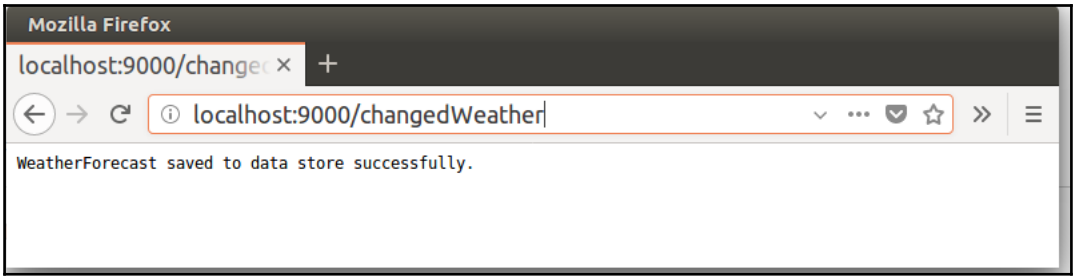


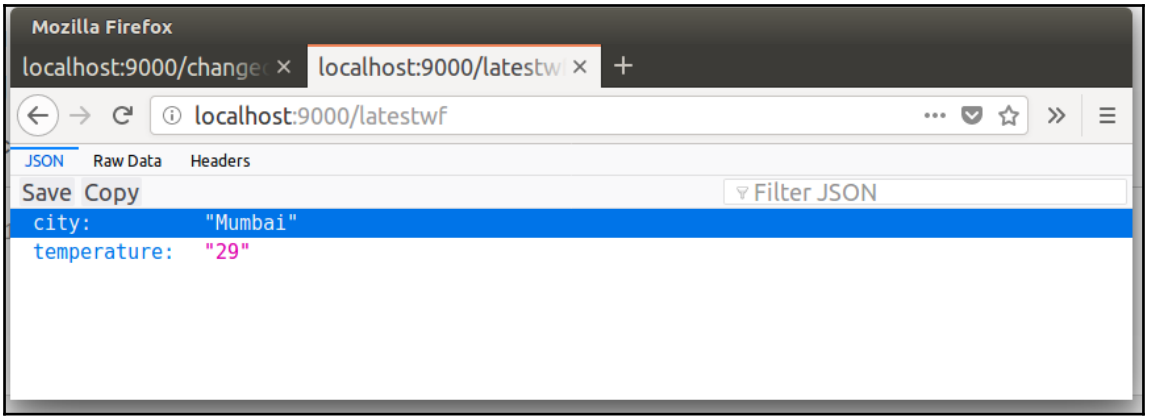

```
rambabuposa@ram:~/Applications/lagom-wf-reactive-system$ sudo sbt
[sudo] password for rambabuposa:
[info] Loading project definition from /home/rambabuposa/Applications/lagom-wf-reactive-system/project
[info] Set current project to play-lagom (in build file:/home/rambabuposa/Applications/lagom-wf-reactive-system/)
>
```

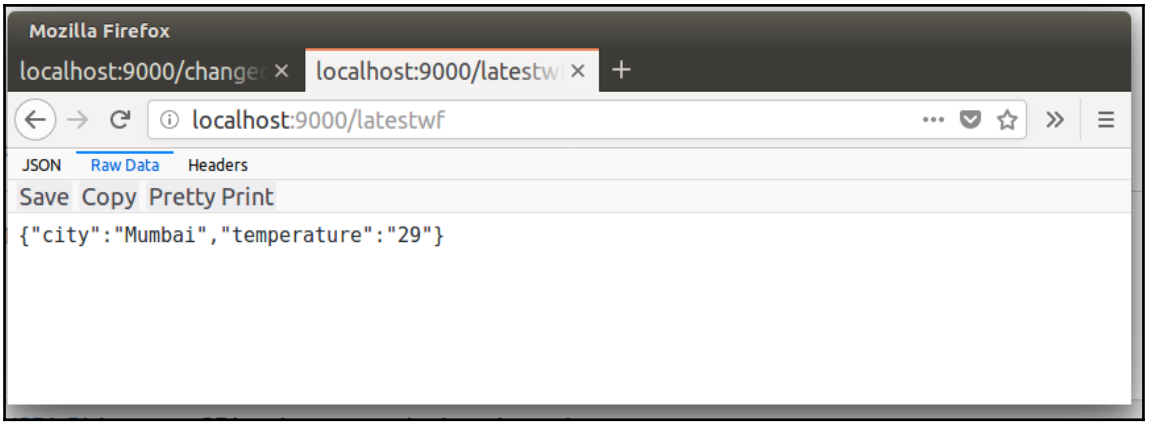
```
rambabuposa@ram: ~/Applications/lagom-wf-reactive-system
> projects
[info] In file:/home/rambabuposa/Applications/lagom-wf-reactive-system/
[info]     lagom-internal-meta-project-cassandra
[info]     lagom-internal-meta-project-kafka
[info]     lagom-internal-meta-project-service-locator
[info]   * play-lagom
[info]     wf-consumer-api
[info]     wf-consumer-impl
[info]     wf-forntned
[info]     wf-producer-api
[info]     wf-producer-impl
>
```

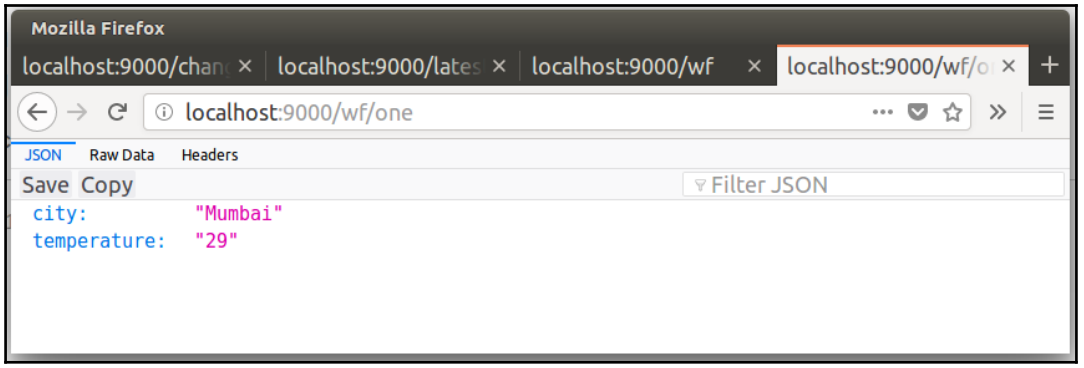
```
Open
version              = "1"
name                 = "wf-producer-impl"
compatibilityVersion = "1"
tags                 = ["1.0.0"]
annotations          = {}
system               = "wf-producer-impl"
systemVersion        = "1"
nrOfCpus             = 0.1
memory               = 402653184
diskSpace            = 200000000
roles                = ["web"]
components = {
  wf-producer-impl = {
    description      = "wf-producer-impl"
    file-system-type = "universal"
    start-command    = ["wf-producer-impl/bin/wf-producer-impl", "-J-Xms134217728", "-J-Xmx134217728", "-Dplay.crypto.secret=77662d70726f64756365722d696d706c"]
    endpoints = {
      "akka-remote" = {
        bind-protocol = "tcp"
        bind-port     = 0
        services      = []
      }
    }
  }
}
```

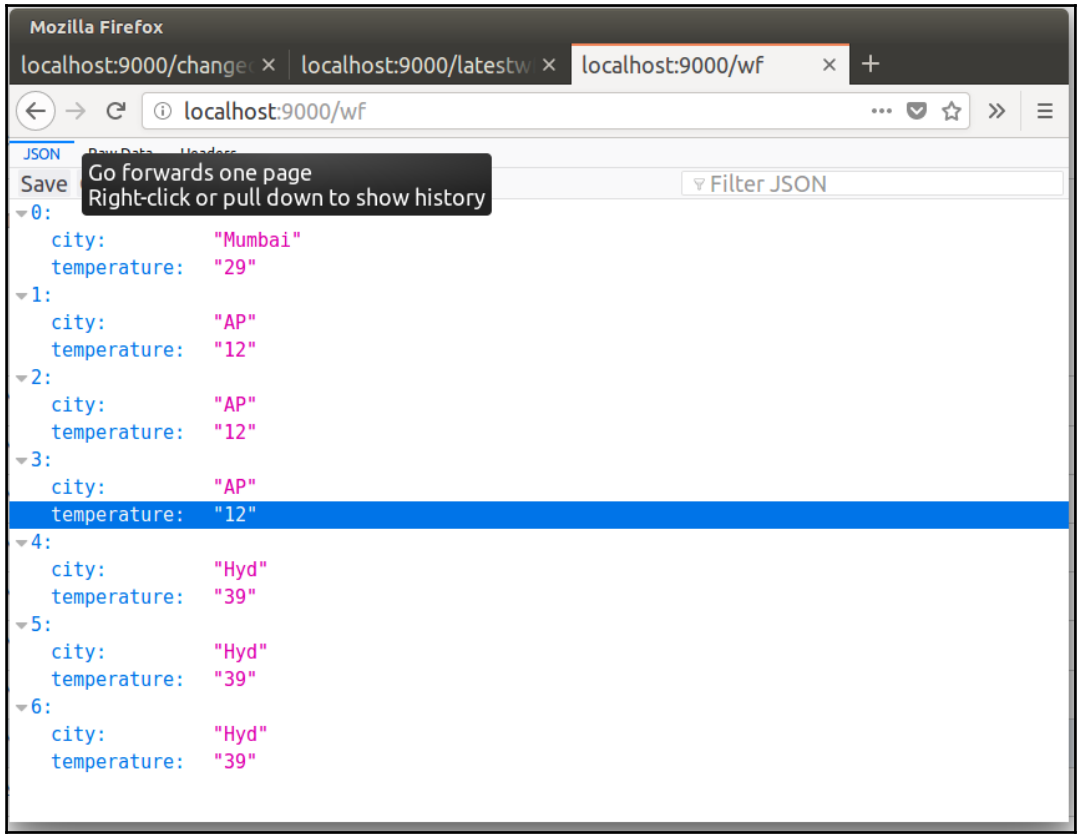



Mozilla Firefox
localhost:9000/changed
localhost:9000/changedWeather
WeatherForecast saved to data store successfully.

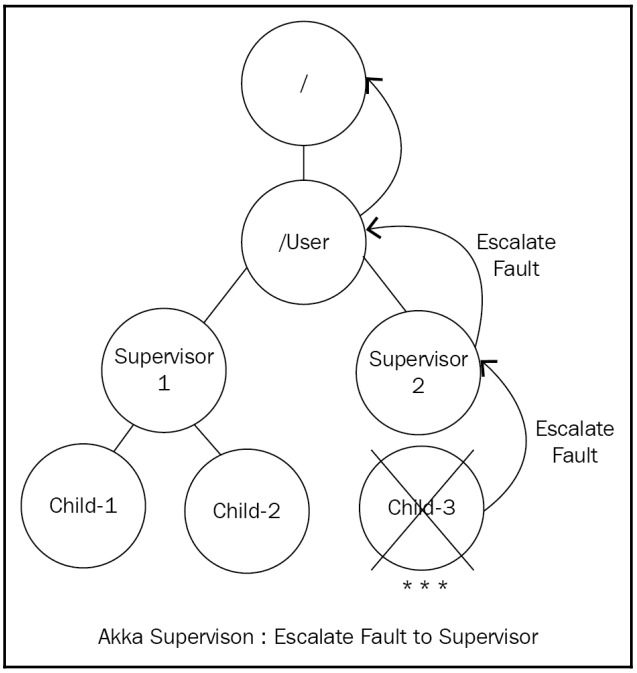

Mozilla Firefox
localhost:9000/changed
localhost:9000/latestwf
JSON
Raw Data
Headers
Save
Copy
Filter JSON
city: "Mumbai"
temperature: "29"


Mozilla Firefox
localhost:9000/changed
localhost:9000/latestwf
JSON
Raw Data
Headers
Save
Copy
Pretty Print
{"city":"Mumbai","temperature":"29"}

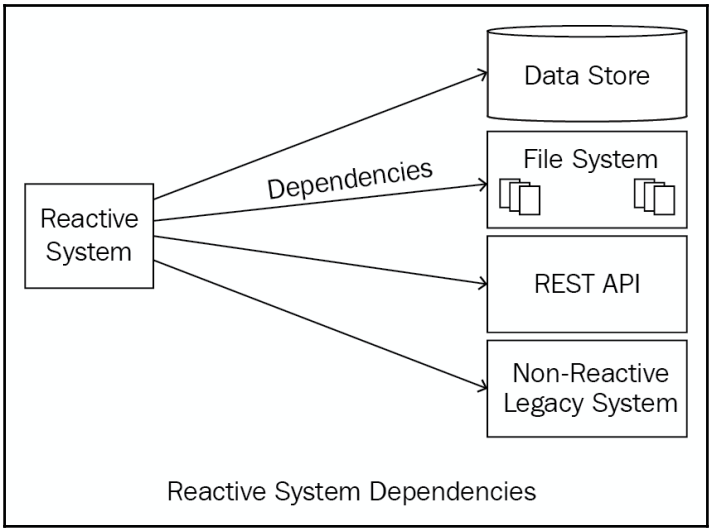

Mozilla Firefox
localhost:9000/chan
localhost:9000/lates
localhost:9000/wf
localhost:9000/wf/o
localhost:9000/wf/one
JSON
Raw Data
Headers
Save
Copy
Filter JSON
city: "Mumbai"
temperature: "29"


Mozilla Firefox
localhost:9000/change
localhost:9000/latestw
localhost:9000/wf
localhost:9000/wf
JSON
Save
Go forwards one page
Right-click or pull down to show history
Filter JSON
0:
city: "Mumbai"
temperature: "29"
1:
city: "AP"
temperature: "12"
2:
city: "AP"
temperature: "12"
3:
city: "AP"
temperature: "12"
4:
city: "Hyd"
temperature: "39"
5:
city: "Hyd"
temperature: "39"
6:
city: "Hyd"
temperature: "39"

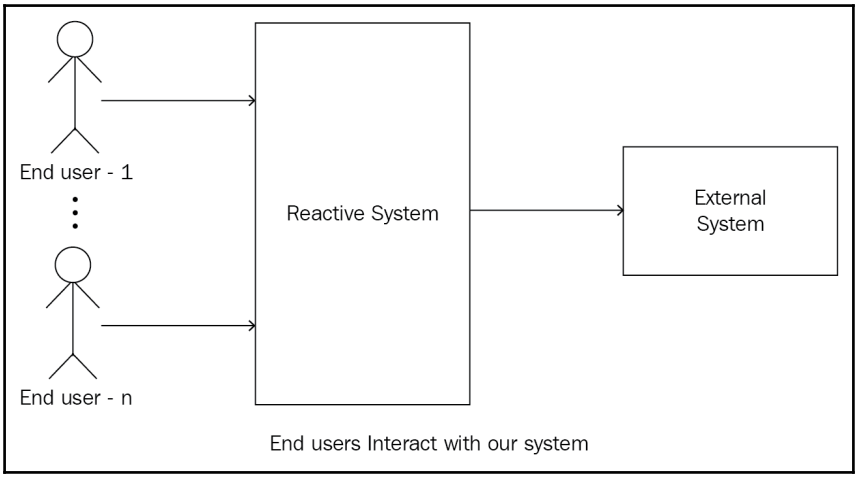
Chapter 12: Reactive Design Patterns and Best Practices



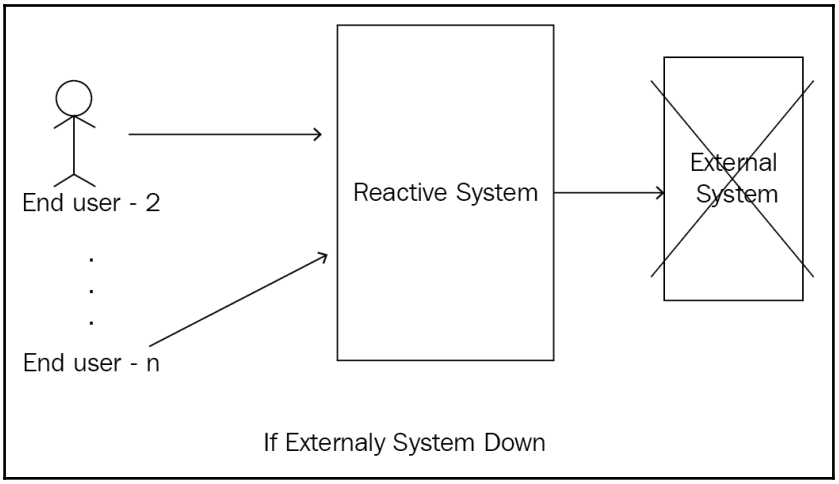
Akka Supervison : Escalate Fault to Supervisor

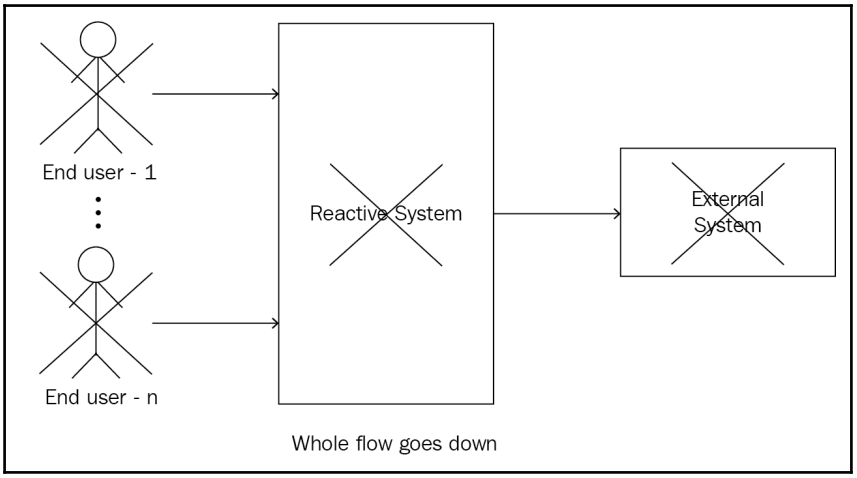


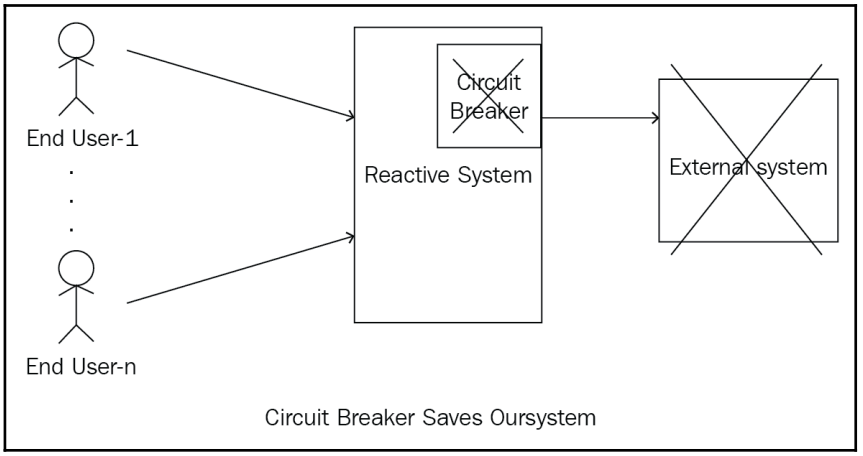
Reactive System Dependencies

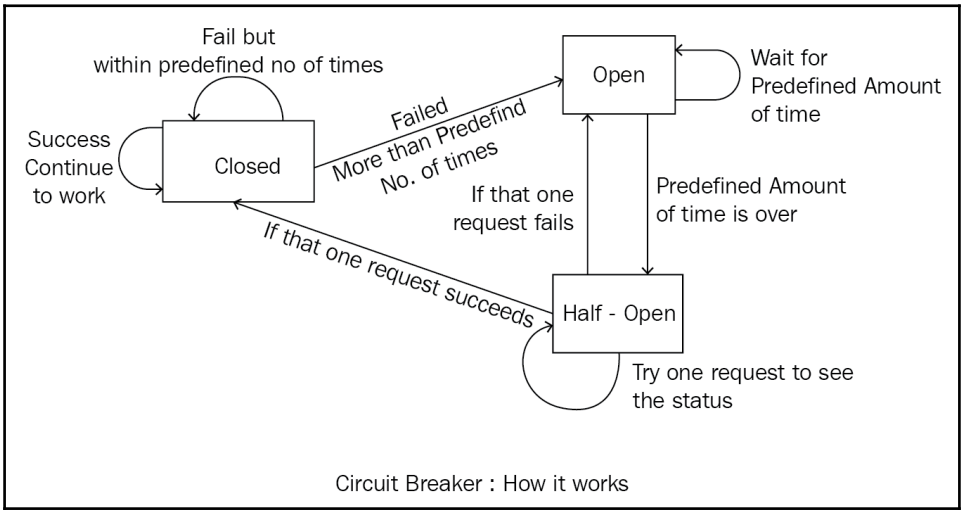


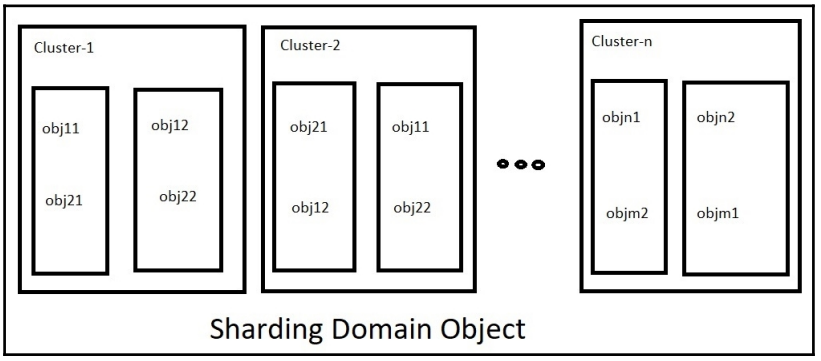
End users Interact with our system

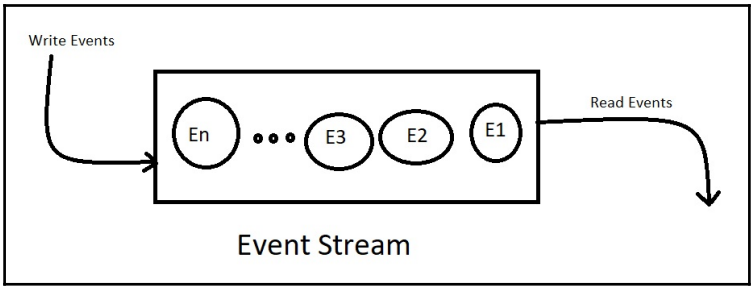

End user - 2
End user - n
Reactive System
External System
If Externaly System Down

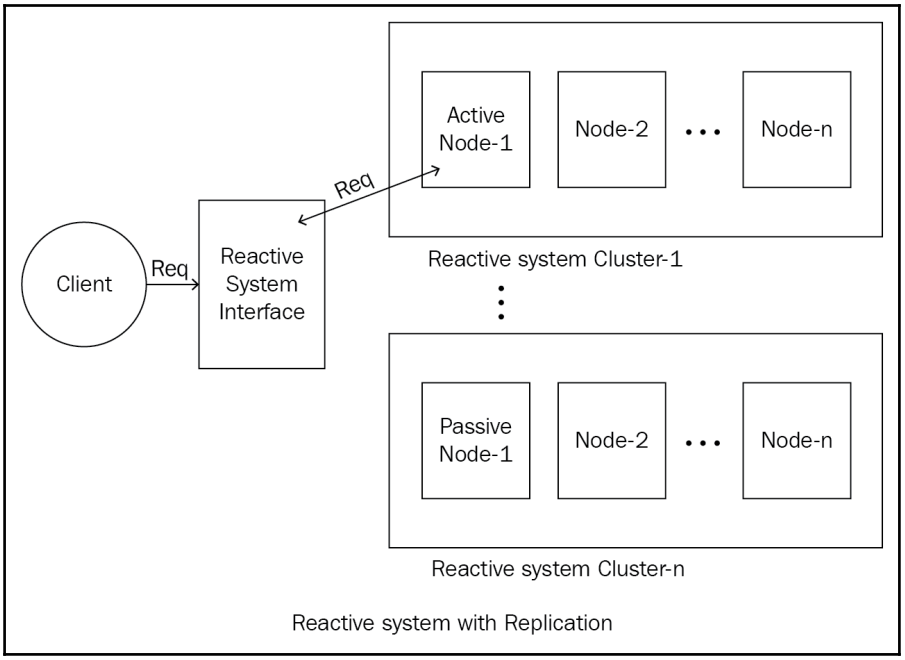

End user - 1
End user - n
Reactive System
External System
Whole flow goes down

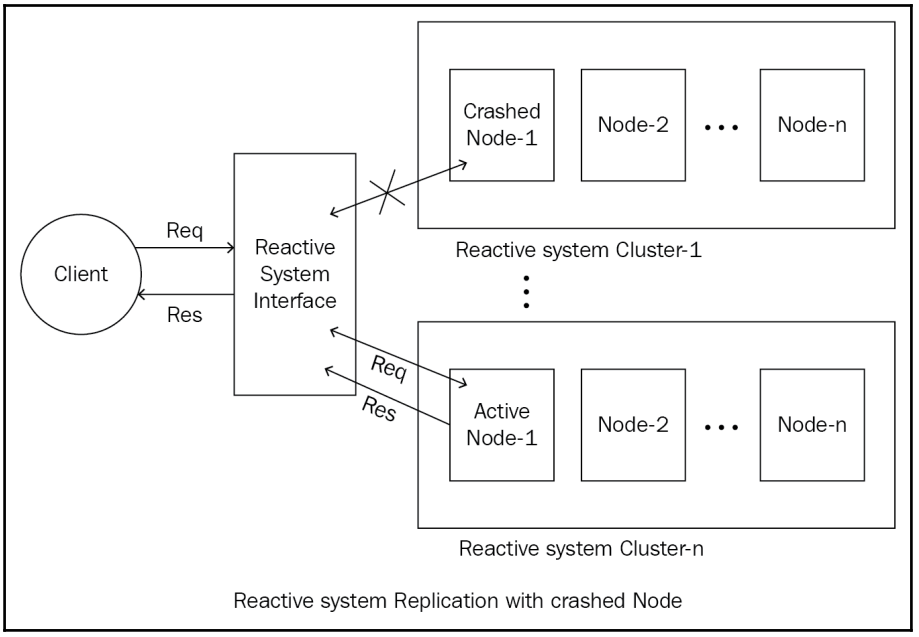

End User-1
End User-n
Circuit Breaker
Reactive System
External system
Circuit Breaker Saves Oursystem


Fail but
within predefined no of times
Success
Continue
to work
Closed
Failed
More than Predefind
No. of times
Open
Wait for
Predefined Amount
of time
If that one
request fails
Predefined Amount
of time is over
If that one request succeeds
Half - Open
Try one request to see
the status
Circuit Breaker : How it works

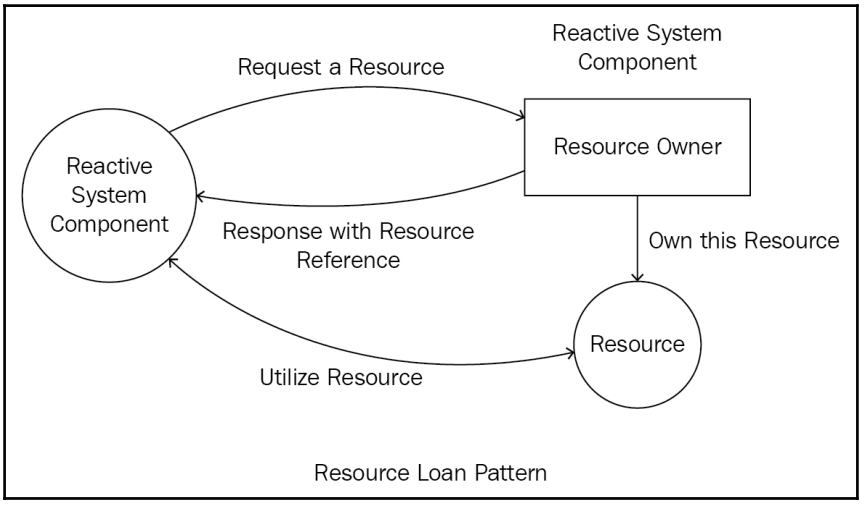

Cluster-1
obj11
obj21
obj12
obj22
Cluster-2
obj21
obj12
obj11
obj22
Cluster-n
objn1
objm2
objn2
objm1
Sharding Domain Object


Write Events
En
E3
E2
E1
Read Events
Event Stream

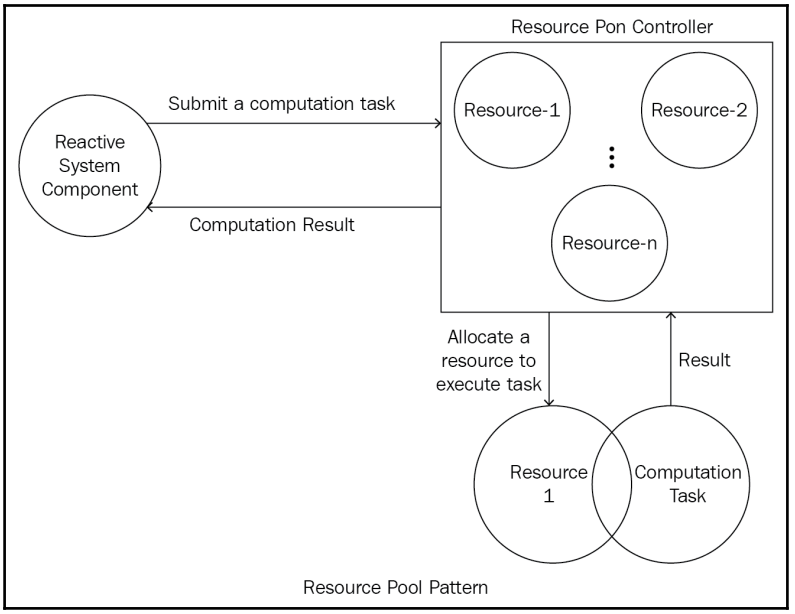

Active Node-1
Node-2
Node-n
Req
Client
Req
Reactive System Interface
Reactive system Cluster-1
Passive Node-1
Node-2
Node-n
Reactive system Cluster-n
Reactive system with Replication

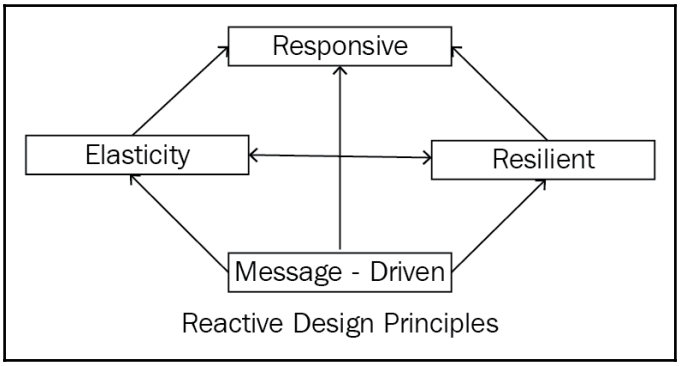


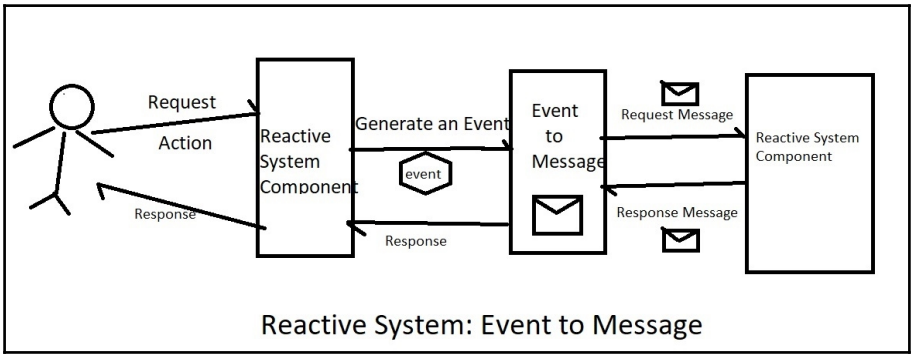
Reactive system Replication with crashed Node

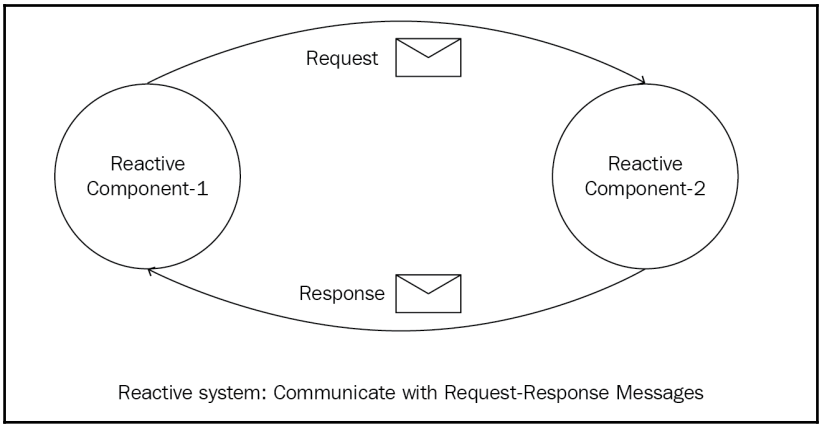


Resource Loan Pattern


Resource Pon Controller
Submit a computation task
Reactive System Component
Computation Result
Resource-1
Resource-2
Resource-n
Allocate a resource to execute task
Result
Resource 1
Computation Task
Resource Pool Pattern

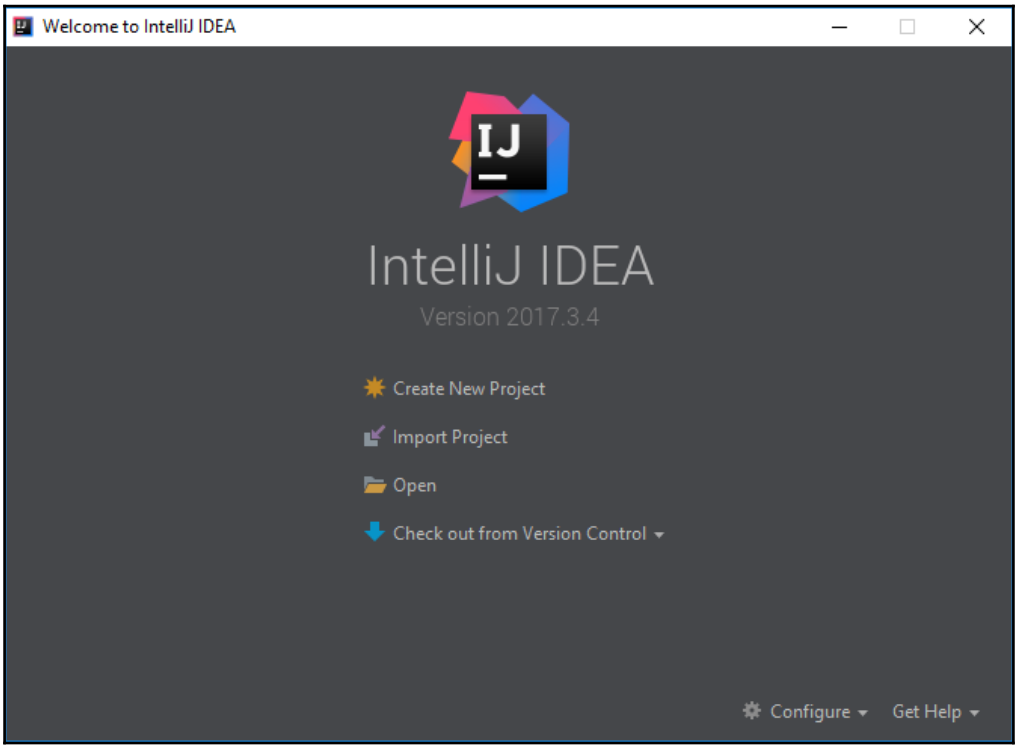

Responsive
Elasticity
Resilient
Message - Driven
Reactive Design Principles

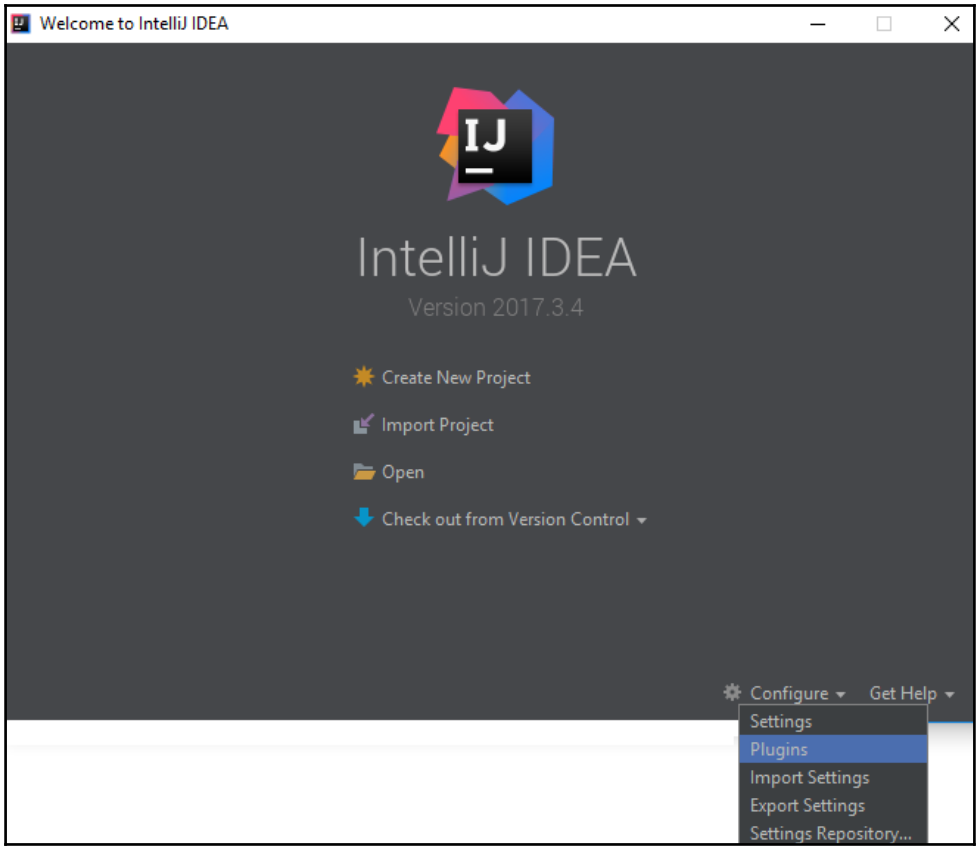

Request
Action
Response
Reactive System Component
Generate an Event
event
Response
Event to Message
Request Message
Response Message
Reactive System Component
Reactive System: Event to Message

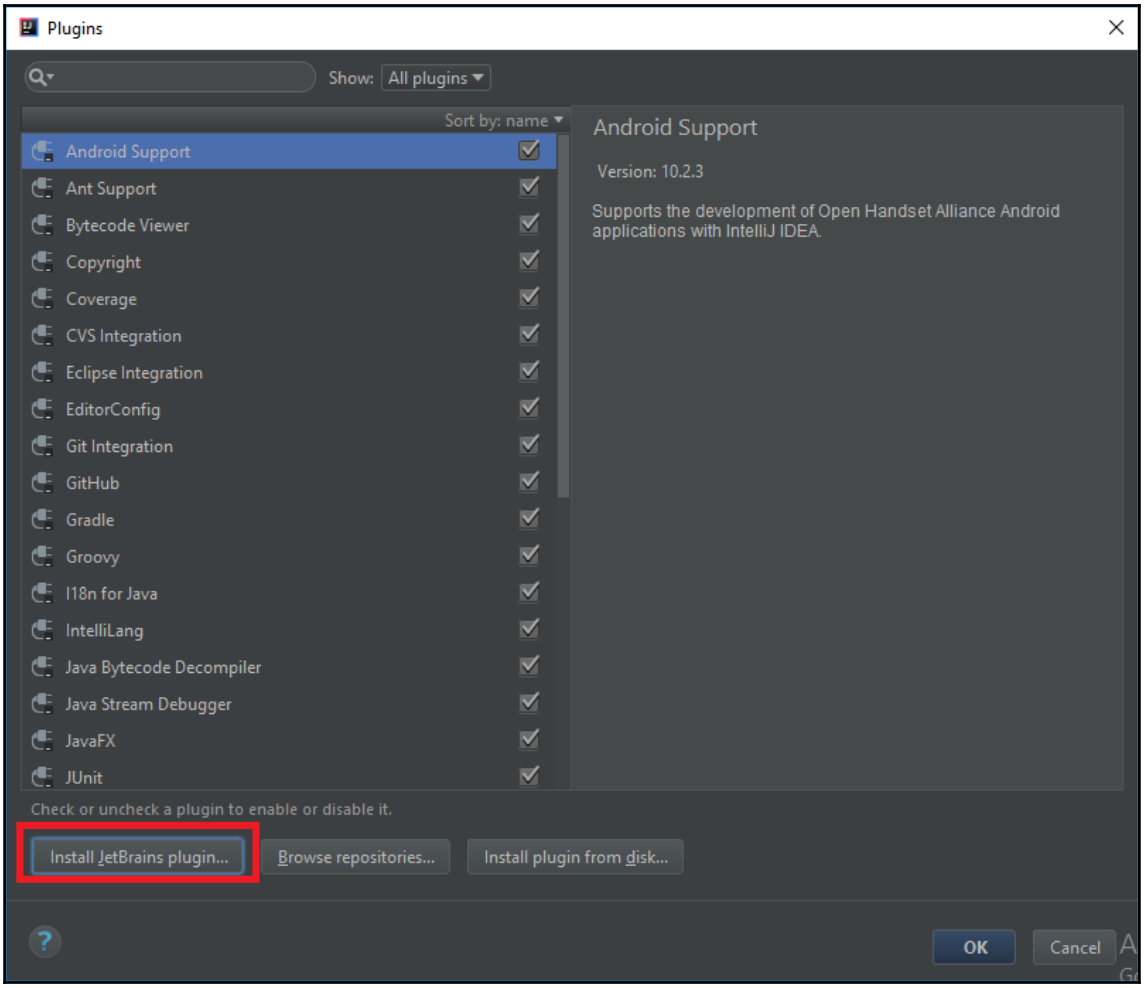


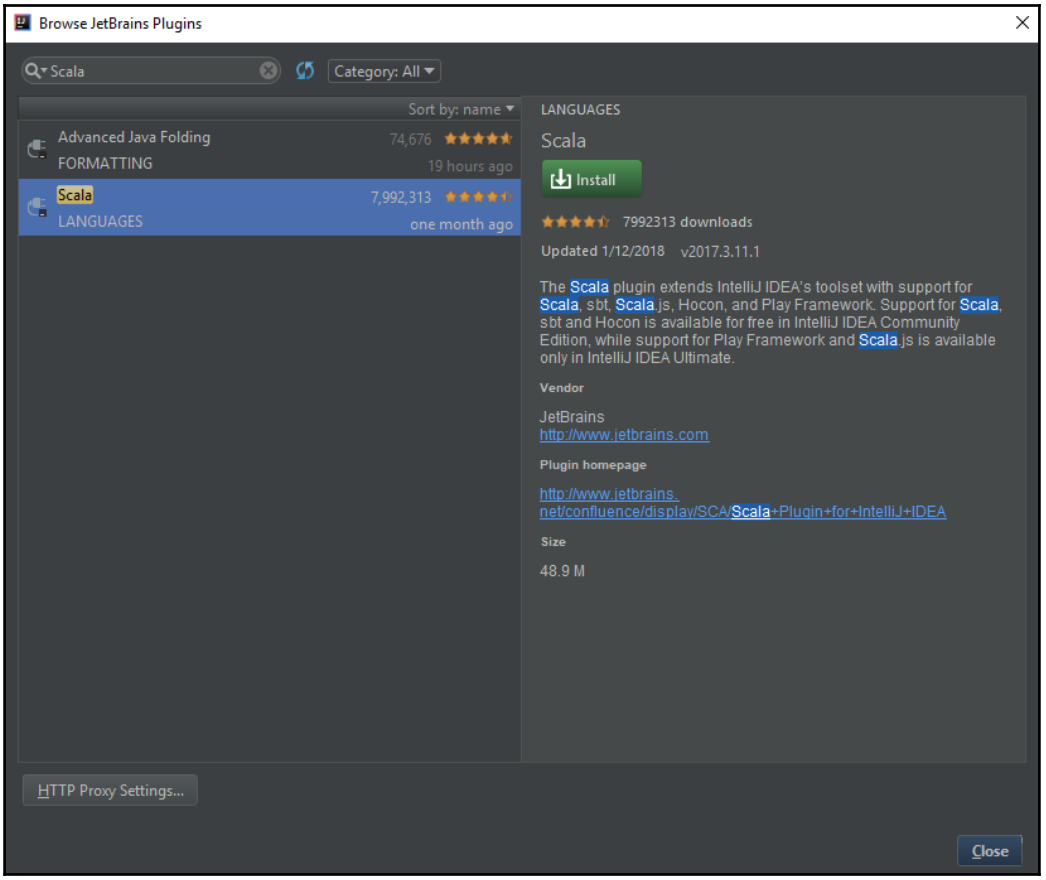
Reactive system: Communicate with Request-Response Messages

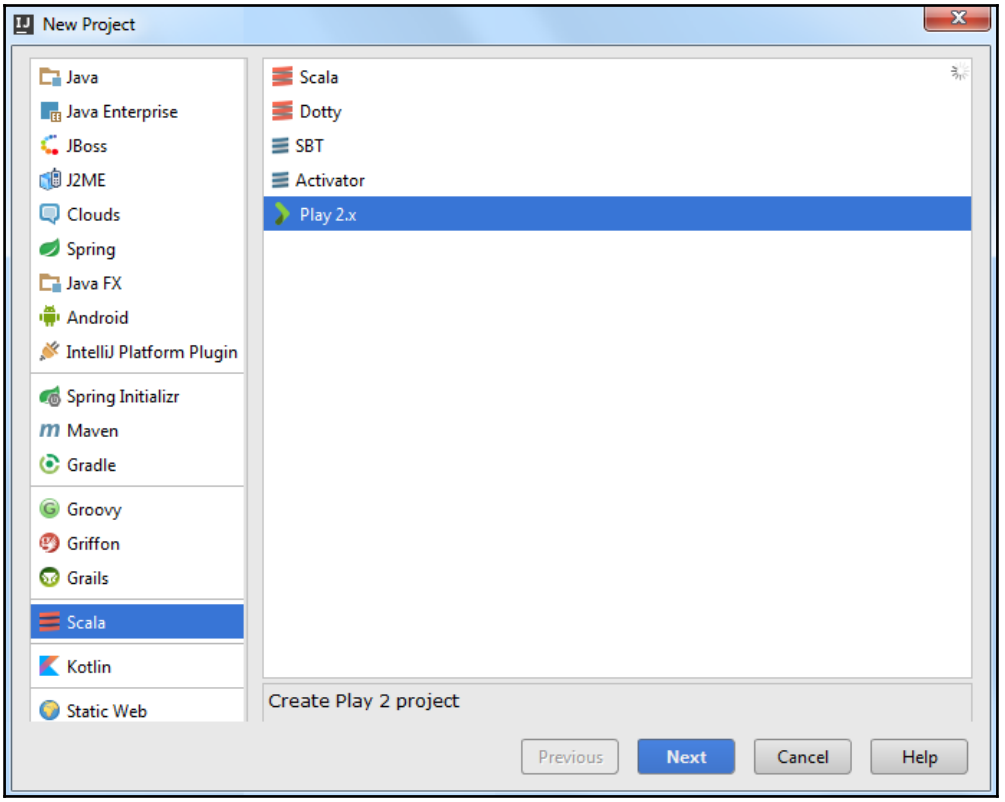
Appendix A: Scala Plugin for IntelliJ IDEA



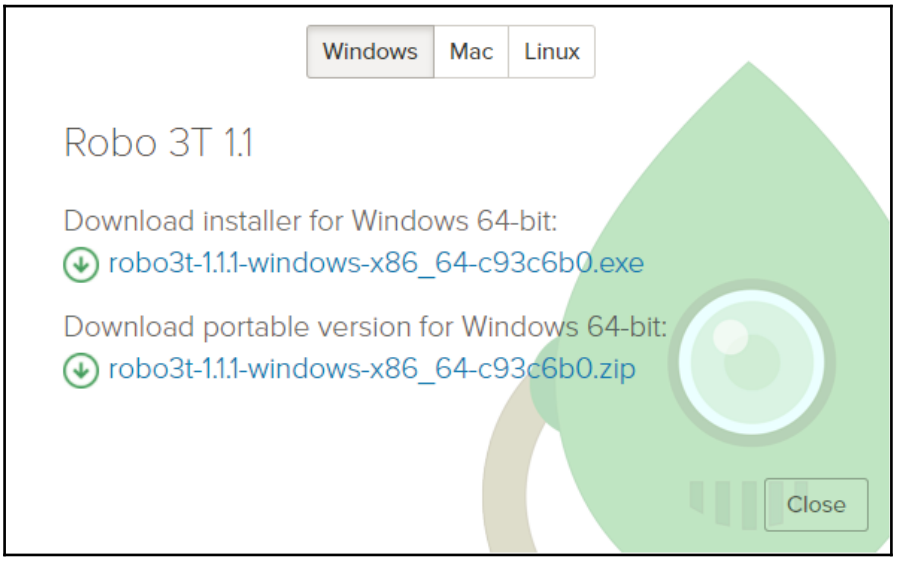

Welcome to IntelliJ IDEA
IntelliJ IDEA
Version 2017.3.4
Create New Project
Import Project
Open
Check out from Version Control
Configure
Get Help
Settings
Plugins
Import Settings
Export Settings
Settings Repository...

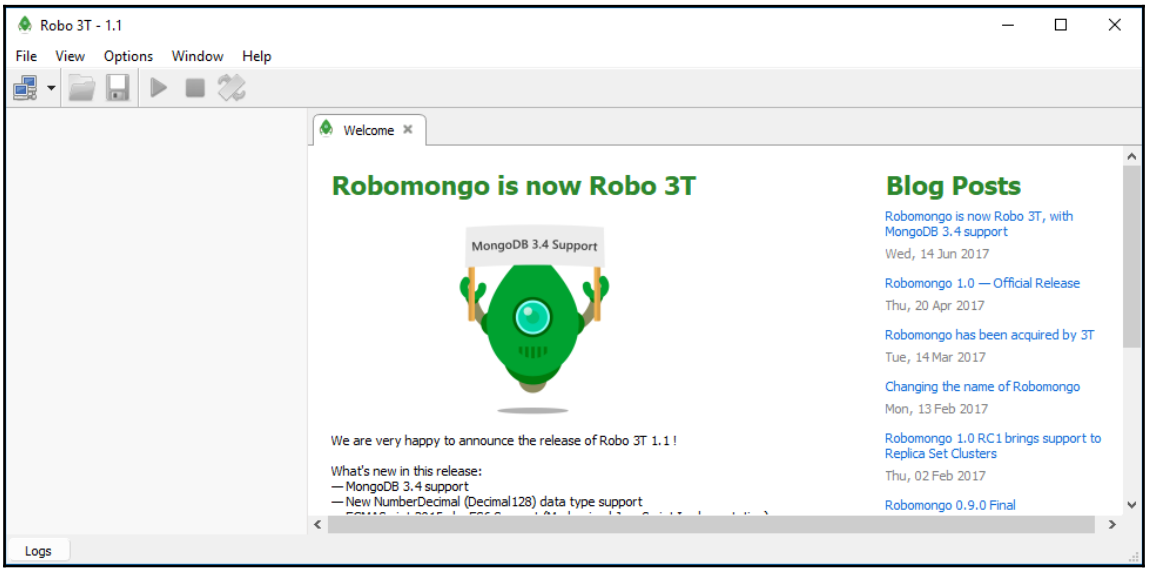

Plugins
Show: All plugins
Sort by: name
Android Support
Ant Support
Bytecode Viewer
Copyright
Coverage
CVS Integration
Eclipse Integration
EditorConfig
Git Integration
GitHub
Gradle
Groovy
I18n for Java
IntelliLang
Java Bytecode Decompiler
Java Stream Debugger
JavaFX
JUnit
Android Support
Version: 10.2.3
Supports the development of Open Handset Alliance Android applications with IntelliJ IDEA.
Check or uncheck a plugin to enable or disable it.
Install JetBrains plugin...
Browse repositories...
Install plugin from disk...
OK
Cancel

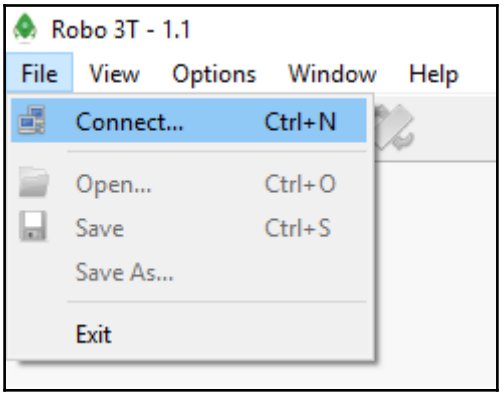

Browse JetBrains Plugins
Scala
Category: All
Sort by: name
Advanced Java Folding
FORMATTING
74,676
19 hours ago
Scala
LANGUAGES
7,992,313
one month ago
LANGUAGES
Scala
Install
7992313 downloads
Updated 1/12/2018 v2017.3.11.1
The Scala plugin extends IntelliJ IDEA's toolset with support for Scala, sbt, Scala.js, Hocon, and Play Framework. Support for Scala, sbt and Hocon is available for free in IntelliJ IDEA Community Edition, while support for Play Framework and Scala.js is available only in IntelliJ IDEA Ultimate.
Vendor
JetBrains
http://www.jetbrains.com
Plugin homepage
http://www.jetbrains.net/confluence/display/SCA/Scala+Plugin+for+IntelliJ+IDEA
Size
48.9 M
HTTP Proxy Settings...
Close

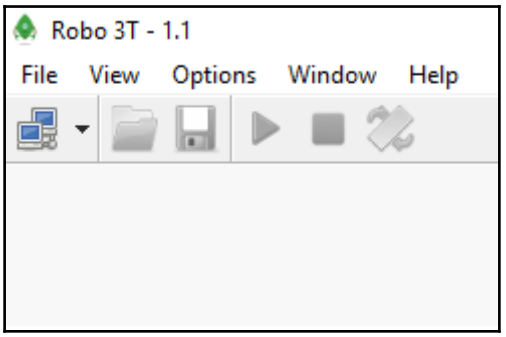

New Project
Java
Java Enterprise
JBoss
J2ME
Clouds
Spring
Java FX
Android
IntelliJ Platform Plugin
Spring Initializr
Maven
Gradle
Groovy
Griffon
Grails
Scala
Kotlin
Static Web
Scala
Dotty
SBT
Activator
Play 2.x
Create Play 2 project
Previous
Next
Cancel
Help

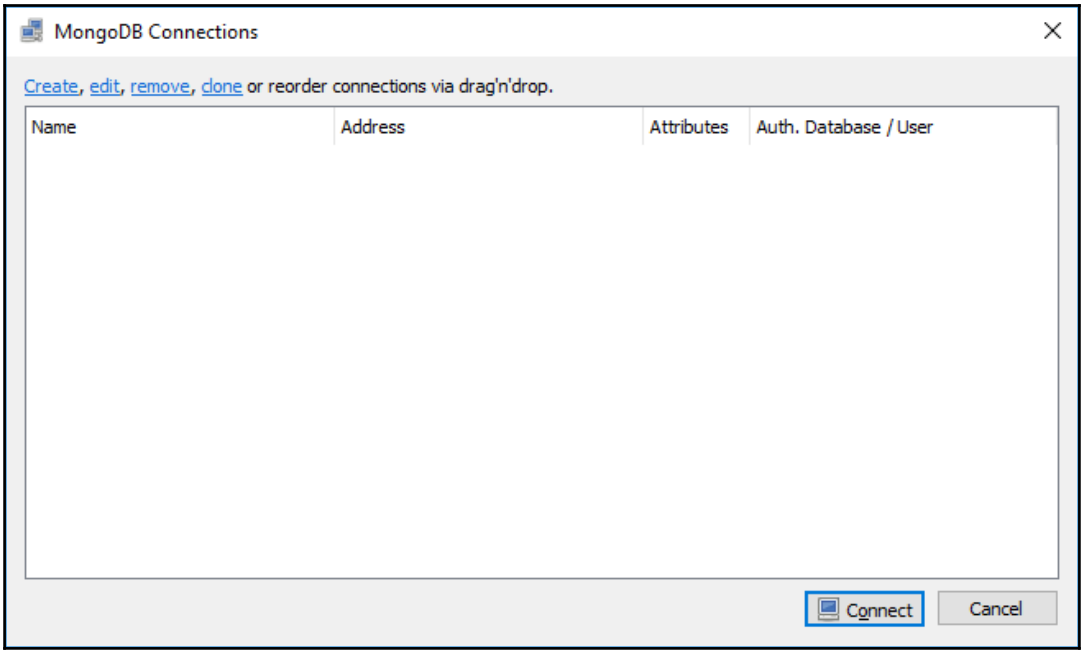
Appendix B: Installing Robomongo

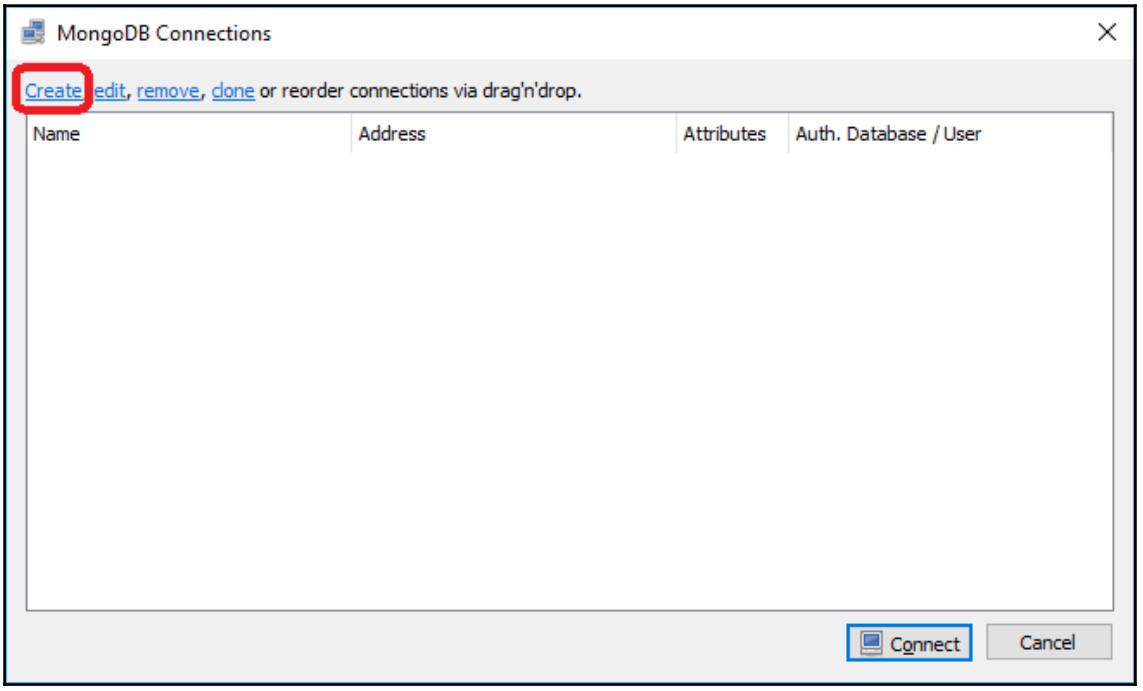


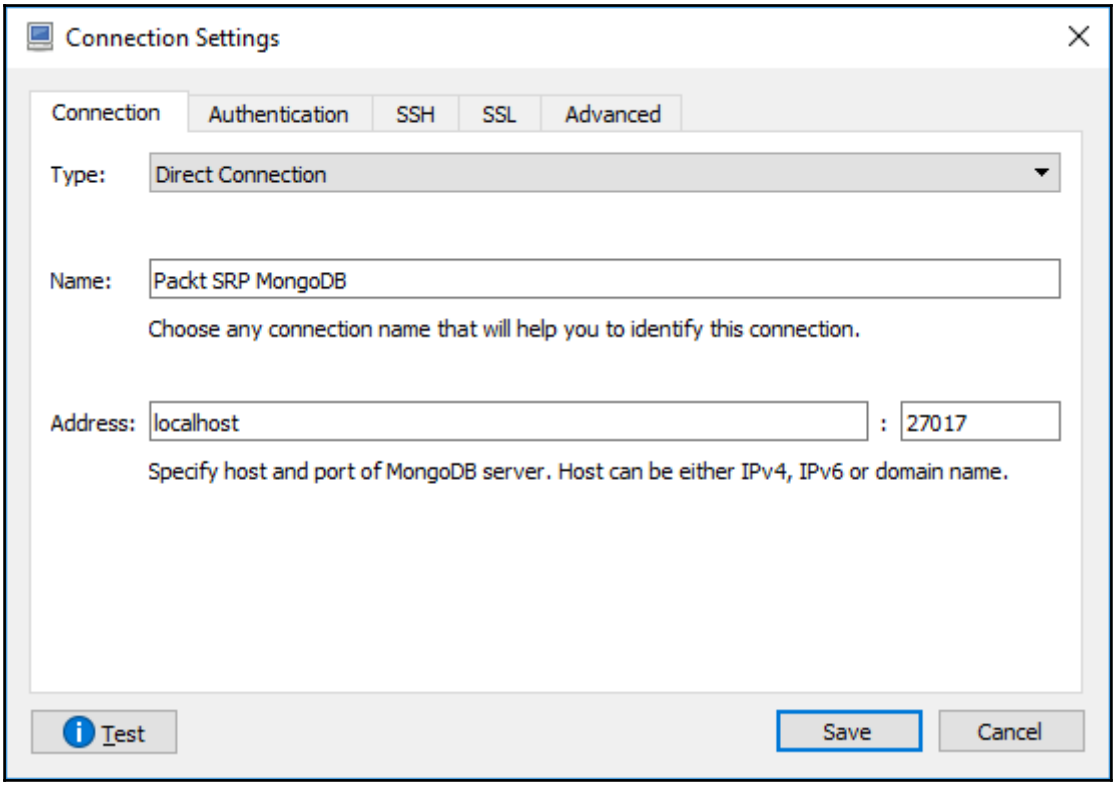



Robo 3T - 1.1
File
View
Options
Window
Help
Connect... Ctrl+N
Open... Ctrl+O
Save Ctrl+S
Save As...
Exit

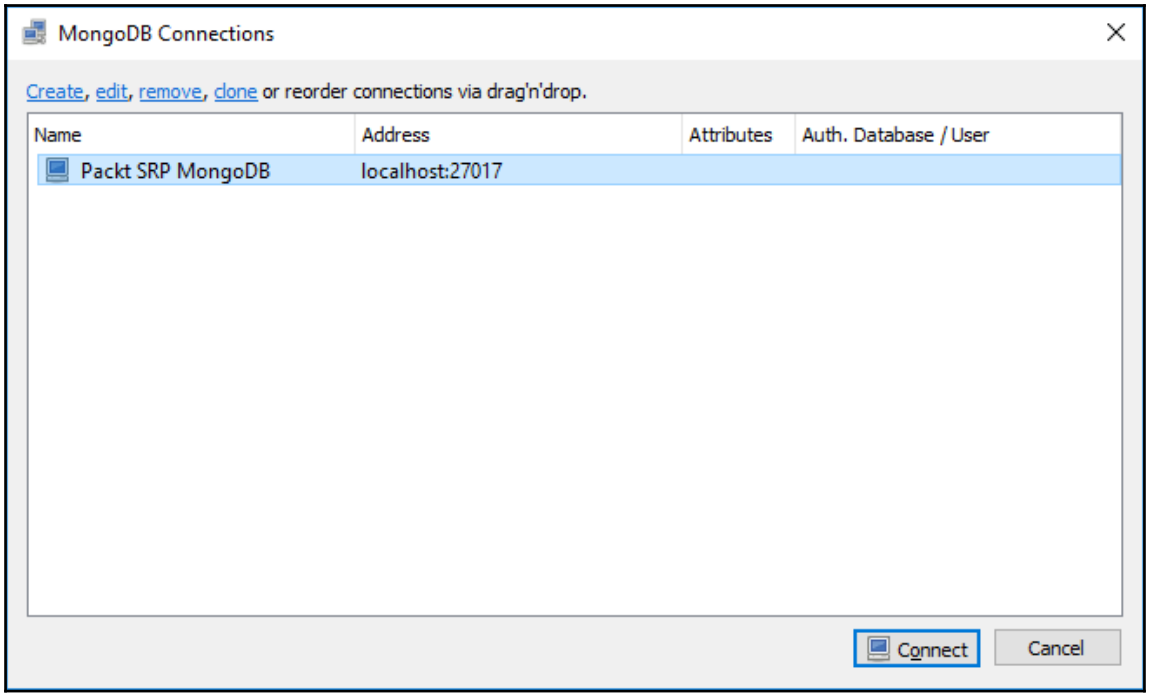

Robo 3T - 1.1
File
View
Options
Window
Help

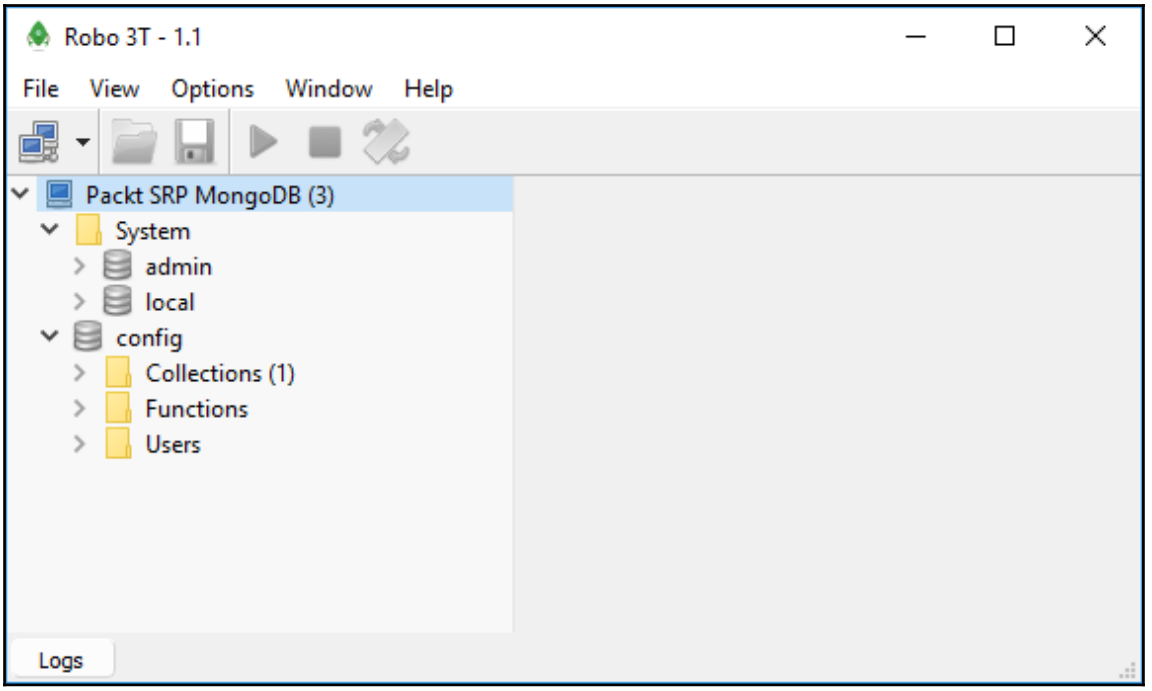

MongoDB Connections
Create, edit, remove, clone or reorder connections via drag'n'drop.
Name
Address
Attributes
Auth. Database / User
Connect
Cancel


MongoDB Connections
Create edit, remove, clone or reorder connections via drag'n'drop.
Name
Address
Attributes
Auth. Database / User
Connect
Cancel


Connection Settings
Connection
Authentication
SSH
SSL
Advanced
Type:
Direct Connection
Name:
Packt SRP MongoDB
Choose any connection name that will help you to identify this connection.
Address:
localhost
:
27017
Specify host and port of MongoDB server. Host can be either IPv4, IPv6 or domain name.
Test
Save
Cancel

>


MongoDB Connections
Create, edit, remove, clone or reorder connections via drag'n'drop.
Name
Address
Attributes
Auth. Database / User
Packt SRP MongoDB
localhost:27017
Connect
Cancel


Robo 3T - 1.1
File
View
Options
Window
Help
Packt SRP MongoDB (3)
System
admin
local
config
Collections (1)
Functions
Users
Logs