

Promises



Promises

Futures are "read-only" async computations

Promises are "controllable" wrappers over a Future

Thread 1:

- creates an empty promise
- knows how to handle result

```
val p = Promise[Int]()  
val future = p.future  
  
future.onComplete {  
  case Success(value) => ...  
  case Failure(ex) => ...  
}
```

Thread 2:

- has a reference to the promise
- fulfills or fails the promise

```
val result = doComputation()  
p.success(result)
```

```
p.failure(new BadException(...))
```

```
p.complete(Try {...})
```

trigger
completion

The diagram illustrates the interaction between Thread 1 and Thread 2. Thread 1 creates a Promise and registers an onComplete handler. Thread 2, which has a reference to the Promise, can trigger its completion by calling success, failure, or complete. An orange line connects the onComplete handler in Thread 1's code to the completion methods in Thread 2's code, with the label 'trigger completion' at the bottom.

Scala rocks

