Collections and Sequences

Kevin Jones

@kevinrjones

Overview

Collections in the standard library

Mapping and filtering collections

Sequences and the problem they solve

Mapping and filtering sequences

Arrays in Kotlin

Array is a class with a type parameter

- Array<Int> etc.

Can create with

- arrayOf
- arrayOfNulls
- Array() function

Primitive Types

Each primitive has its own array type

- IntArray
- ByteArray
- CharArray
- etc.

Creating Collections

listOf, setOf, mapOf

arrayListOf, hashSetOf, hashMapOf

mutableListOf,

Etc.

Functional Style and Collections

Many benefits of using functional style with collections

Can use library functions

Simplifies code

May look odd if you are not used to it

Predicates

all

any

count

find

Essential Functions

filter

map

Filter

Transforms collections

Filters out unwanted items

Similar to 'where' in SQL

Map

Transforms items

Similar to 'select' in SQL

Problem with Standard Collections

Functions such as filter and map create lists

Fine if lists are small

Not good if lists are massive

Instead use Sequences

Example of Creating a Sequence

```
val titles = collection
             .asSequence()
             .filter {it -> it.title.startsWith("C")}
             .map { m-> m.title}
```

Terminal and Non-terminal Operations

Sequences are lazily evaluated

Evaluation starts when using terminal operation

Summary

Collections in the standard library

- Array
- List
- Map
- Set

Immutable collections

- List
- Map
- Set

Mutable versions

- MutableList
- etc

Summary

Have predicates

- any
- all
- count

Can map and filter

- filter
- map

Summary

Sequences solve some problems

- Large collections
- Expensive operations

Terminal and non-terminal operations

- Don’t execute until terminal operation called
- sum
- take
- etc

Nullability