Tìm hiểu các cấu trúc dữ liệu trong Dart

Các cấu trúc dữ liệu từ đơn giản đến phức tạp trong Dart như enum, list, map, set, hashmap, hashset

Trong phần này chúng ta sẽ tìm hiểu một số cấu trúc dữ liệu cơ bản đã triển khai sẵn trong ngôn ngữ lập trình Dart:

Cấu trúc dữ liệu liệt kê enum

Trong Dart đây là một loại lớp đặc biệt, biểu diễn một tập hợp cố định các hằng số. Để tạo ra một enum dùng từ khóa enum khai báo các phần tử liệt kê theo tên cách nhau bởi , ví dụ:

enum UserGroup {guest, member, admin}

Enum rất tiện lợi dùng với câu lệnh switch

var user\_group = UserGroup.admin;

switch (user\_group) {

case UserGroup.admin:

print('Quản trị hệ thống');

break;

case UserGroup.guest:

print('Khách');

break;

default:

}

Cấu trúc dữ liệu Iterable

Iterable là một lớp generic biểu diễn tập hợp dữ liệu mà có thể duyệt qua hết phần tử này đến phần tử khác. Nghĩa là nó hỗ trợ moveNext() để đi đến phần tử tiếp theo, lấy dữ liệu phần tử hiện tại bằng iterator.current

Thường thì Iterable được tạo ra, liên kết với một loại kiểu dữ liệu tập hợp khác như List, Map ...Xem các loại cấu trúc dữ liệu này để tìm hiểu về Interable

Duyệt qua các phần tử Iterable

//Sinh ra Iterable chứa 100 phần tử số từ 0 - đến 99

var iterable = new Iterable.generate(100);

for (var item in iterable) {

print(item);

}

//0

//1

...

//99

Hoặc duyệt qua bằng forEach

iterable.forEach((f) {

print(f);

});

**Các phương thức / thuộc tính hay dùng trên mảng - danh sách**

|  |  |
| --- | --- |
| **Phương thức** | **Sử dụng** |
| isEmpty | Thuộc tính kiểm tra xem mảng rỗng |
| isNotEmpty | Thuộc tính kiểm tra xem mảng không rỗng |
| length | Thuộc tính trả về số lượng phần tử mảng |
| first | Thuộc tính trả về phần tử đầu tiên, tương đương với [0], lỗi nếu mảng rỗng |
| last | Thuộc tính trả về phần tử đầu cuối |
| forEach() | Duyệt qua các phần tử  iterable.forEach((f) {  print(f);  }); |

Cấu trúc dữ liệu Danh sách - Mảng - List

Trong Dart, danh sách (cũng là mảng) được định nghĩa từ lớp generic List, nó chứa một tập hợp các dữ liệu - mỗi dữ liệu trong List là gọi là phần tử, vị trí của nó xác định bằng chỉ số bắt đầu từ 0, truy cập đến mảng (danh sách) dùng ký hiệu [] chứa chỉ số phần tử.

Có 2 loại List, đó là loại mà số phần tử có thể thay đổi và loại list có số phần tử cố định

**Khởi tạo một mảng cố định**

//Khai báo mảng cố định 2 phần tử

var listState = new List(2);

//Khởi tạo các phần tử trong mảng

listState[0] = 'on';

listState[1] = 'off';

//Nếu truy cập listState[2] sẽ lỗi- vì List chỉ có 2 phần tử

print(listState); //[on, off]

**Khởi tạo một mảng thay đổi số phần tử được**

Nếu khi khởi tảo mà không chỉ ra số lượng phần tử thì nó là mảng thay đổi số phần tử được, lúc này có thể áp dụng các hàm thêm, bớt phần tử sẽ nói phía dưới

//Khai báo mảng thay đổi được

var dow = new List();

//Thêm các phần tử vào mảng

dow.add('Monday');

dow.add('Tuesday');

dow.add('Thursday');

print(dow); //[Monday, Tuesday, Thursday]

//Xóa phần tủ cuối cùng

dow.removeLast();

print(dow); //[Monday, Tuesday]

Nếu muốn tạo ra mảng thay đổi được, và khởi tạo luôn dữ liệu ở phần khai báo thì dùng ký hiệu []

//Khởi tạo mảng với 2 phần tử

var group = ['member', 'admin'];

group.insert(0, 'guest'); //Chèn phần tử vào vị trí 0

print(group); //[guest, member, admin]

**Các phương thức / thuộc tính hay dùng trên mảng - danh sách**

Ngoài các phương thức - thuộc tính giống như Iterable chú ý thêm:

|  |  |
| --- | --- |
| reversed | Trả về một đối tượng Iterable chứa các phần tử mảng theo thứ tứ ngược lại |
| add() | Thêm một phần tử vào cuối add(element) |
| insert() | Chèn một phần tử vào mảng ở vị trí i insert(i,element) |
| insertAll() | Chèn một một Iterable bắt đầu từ vị trí i: insertAll(i,iterable) |
| remove() | Xóa bỏ phần tử đầu tiền tìm thấy có giá trị chỉ ra remove(data) |
| removeAt() | Xóa bỏ phần tử ở vị trí i removeAt(i) |
| removeLast() | Xóa bỏ phần tử cuối |
| removeRange() | Xóa bỏ phần tử từ vị trí start đến vị trí end removeRange(start, end) |

Cấu trúc dữ liệu - Map - Ánh xạ

Đây là kiểu tập hợp dữ liệu mà mỗi phần tử biểu diễn theo cặp key:value

Các phần tử của Map được truy cập bằng ký hiệu [] chứa key dạng map[key]

Khởi tạo Map có thể dùng toàn tử new hoặc khởi tạo luôn một số phần tử bằng {}

//Tạo một map, khới tạo luôn 3 key - nam, age, score

var student = {

'name':'Abc',

'age': 22,

'score': 'A'

};

student['Subject'] = 'Abc'; //Thêm một phần tử

print(student['name']); //Truy cập phần tử

Cũng hoàn tạo tạo ra Map từ toán tử new

var student = new Map();

student['age'] = 22;

student['name'] = 'StudentA';

student['score'] = 'A';

//Duyệt qua các phần tử cách 1

student.forEach((key, value) {

print('$key : có giá trị $value');

});

//Duyệt qua tất cả các phần tử Map (cách 2)

for (var key in student.keys) {

print('$key : ${student[key]}');

}

**Các phương thức / thuộc tính hay dùng trên Map**

Ngoài các phương thức - thuộc tính giống như Iterable xem ở trên chú ý thêm:

|  |  |
| --- | --- |
| addAll | Thêm các phần tử từ một map khác addAll(othemap) |
| clear | Làm rỗng Map clear(); |
| containsKey | containsKey(key) kiểm tra phần tử với key tồn tại |
| remove() | remove(key) xóa phần tử khỏi map |

Tập hợp - Set

Tập hợp như tên gọi là là tập hợp các phần tử, đảm bảo sao cho mỗi phần tử chỉ được xuất hiện 1 lần.

Khởi tạo một tập hợp bằng toán tử new với cú pháp

var s = Set();

Nó có các phương thức và cách duyệt qua phần tử giống phần trình bày Iterable ở trên.

Để thêm một phần tử vào tập hợp dùng hàm add(ele); để loại bỏ phần tử dùng hàm remove(ele);, kiểm tra xem có chứa phần tử bằng hàm contains(ele);

Một số cấu trúc dữ liệu phức tạp

Từ các cấu trúc dữ liệu trên, Dart xây dựng thêm các loại cấu trúc phức chứa trong thư viện dart:collection

Một số loại như: HashMap, HashSet, LinkedList ...

Trước khi sử dụng các cấu trúc dữ liệu này cần nạp thư viện bằng lệnh sau ở đầu file

import 'dart:collection';

Hàng đợi - Queue

Với hàng đợi Queue thì chỉ tương tác với cấu trúc này ở đầu hàng đợi và cuối hàng đợi (thêm / bớt).

|  |  |
| --- | --- |
| add, addLast | Thêm phần tử vào cuối hàng add(ele); hoặc addLast(ele) |
| addFirst | Thêm phần tử vào đầu hàng addFirst(ele); |
| removeFirst() | Xóa phần tử đầu tiên |
| removeLast() | Xóa phần tử cuối cùng |
| remove() | Xóa 1 phần tử remove(ele) |

var q = new Queue();

q.add('A');

q.add('B');

q.addFirst('A0');

q.addLast('B0');

print(q);

q.removeFirst();

q.removeLast();

q.remove('B');

Xem thêm Iterable để xem cách duyệt qua một cách tương tự.

HashMap

Sử dụng và ý nghĩa giống Map (khởi tạo new HashMap()), có có điều các đối tượng làm key phải cung cấp sự so sánh == thông qua giá trị .hashCode

**HashMap** sử dụng mã hash của đối tượng key để so sánh, tìm kiếm phần tử nên HashMap nhanh hơn khi dữ liệu key lớn

HashSet

Sử dụng và ý nghĩa giống Set (khởi tạo new HashSet()), so sánh == thông qua giá trị .hashCode

LinkedList

Sử dụng và ý nghĩa giống List (khởi tạo new LinkedList()), chỉ có điều nó lưu trữ dữ liệu theo cách tối ưu để duyệt qua, tìm kiếm nhanh, tốt để tương quản lý dữ liệu, như chèn và xóa một lượng lớn phần tử