|  |  |  |
| --- | --- | --- |
| STT | Tác dụng | Phím |
| 1 | Tạo cửa sổ mới. | Ctrl + Shift +N |
| 2 | Tạo 1 dự án project dart mới | Ctrl + Shift +P |
| 3 | Tạo 1 class StatelessWidget | stl enter |
| 4 | Tạo 1 class StatefulWidget | stf enter |
| 5 | Căn đều hai bên | Thêm const Spacer(); hai bên/ Thêm const Spacer(flex:2); |

###### BUỔI 1: DART

**Chương 1 : Cơ bản**

1.Khái niệm.

* Flutter là 1 framework(là 1 bộ thư viện đã được viết sẵn) hay một bộ SDK tạo bởi ngôn ngữ: C, c++,Dart - viết bởi google, dùng để:
* Phát triển ứng dụng di động chạy đa nền tảng
* Phát triển web
* Bin : chứa file .dart , file chứa hàm main
* Lib: chứa file thư viện sử dụng trong project
* Pubspec: khai báo thư viện cộng đồng

1. Nguyên tắc lập trình:

* Tất cả các tên phải có ý nghĩa, dễ hiểu.
* Tên file: Viết thường, cách nhau bởi dấu ‘\_’. VD login\_screen.dart
* Tên class: iết hoa chữ cái đầu. Vd LoginScreen.
* Tên biến và tên hàm đặt tên có ý nghĩa , viết thường chữ cái đầu, còn các chữ sau viết hoa. VD buttonColor.
* Tên hằng viết hoa toàn bộ và cách nhau dấu ‘\_’.
* Viết code rõ ràng dễ đọc dễ hiểu bằng việt đặt tên có ý nghĩa , ít sử dụng commnet
* Tên trường trong Map : Viết thường cách nhau bởi dâu ‘ \_’.
* Viết 1 lần sửu dụng nhiều nơi(function, widgels, class).
* Hàm là một phương thức được viết ra để thực thi chức năng nào đó gtrong chương trình hay nói cách khác là cài đặt chức năng cho đối tượng
* Hằng final có thể khai báo giá trị của một biến khác còn const thì không.

1. Kiểu dữ liệu

* Static: int, double,String,bool, StringBuffer, RegExp, List,Map,Set,Set,DateTime,..
* Dynamic: var(cùng giá kiểu với biến ban đầu), dynamic(Khác kiểu với biến ban đầu )(Có thể biểu diện bất kể dữ liệu nào).Không nên lạm dụng sẽ bị lỗi
* $tenBien : chuyển int sang String or tenBien.toString() còn hàm thêm dấu ngoạc nhọn bao hàm và thêm kí tự $ trước nó.
* Flutter không nhận giá trị null.Để nhận giá trị null thì thêm ? ngay sau kiểu dữ liệu. VD int? A;
* Thêm required: bắt buộc nhập
* VD: int tinhHieu({required int? a, required int?b)}){
  + - return a!-b!;
    - }
* Return : dùng khi tạo hàm lẹnh dùng để kết thúc hàm và trả về kết quả.
* Break: thường dùng trong vòng lặp và switch dùng để thoát vòng lặp.
* Continue: thường dùng trong vòng lặp.Khi gặp nó tiếp tục vòng lặp mới luôn.
* Map<String,dynamic > map1 = { “name”: “Dart”, “age”: 21};
* =>tenBien <=> return tenBien;

**Chương 2: OOP**

1. **Đối tượng**

* Là các đối tượng trong thế giới thực mà có thể mô tả thông qua thuộc tính và hành riếng của nó.
* Class:
* Thuộc tính
* Phương thức
* Hàm khởi tạo: chỉ có một hàm khởi tạo duy nhất.

-Lấy ngày hiện tại : DateTime.Now();

* Private : cục bộ bên ngoài không thể truy cập được. Chỉ sửu dụng được trong class.
* Có dấu \_ ở trước tên biến VD: String? \_phoneNumber;
* Không nên cho vào hàm khởi tạo vì nó không được công khai
* Nên dùng setter getter để lấy và truyền.
* String get phoneNumber =>this.\_phoneNumber; = String getPhoneNumber (){

return \_phoneNumber;

}

* “\_” trước biến hay hàm, nghĩa là private, tức là chỉ sử dụng được trong class đó.
* Khi khao báo trong class là 1 biến final thì hàm khởi tạo bắt buộc phải có biến này nếu không sẽ báo lỗi.Vì biến final cần được khởi tạo giá trị trước.
* Required đặt trước biến trong hàm thì khi sử dụng nếu không truyền giá trị vào biến đó chắc chắn sẽ lỗi.

1. Đặc tính hướng đối tượng.

* Tính đóng gói: class, setter, getter, private
* Tính kế thừa: extends; :super
* Tính trừu tượng: abtract : có thể có các biến hoặc hàm trừu tượng.Không có thân hàm.
* Tính đa hình

1. Try catch,..

Int makeNumberUp(int value) => value ++;

* Stream<int>.preodic(Duration(second:2), makeNumberUp()); : sau 2s xuất giá trị value tăng 1 lần.
* Trong Dart, chúng ta có ba cách để xử lý các giá trị có thể null:
* Sử dụng dấu chấm hỏi (**?**) để đánh dấu một biến là có thể null. Ví dụ: **int? x = null;**. Biến này có thể chứa một giá trị số nguyên hoặc null.
* Sử dụng dấu chấm than (**!**) để khẳng định rằng một biến không null. Ví dụ: **int x = myNullableValue!;**. Biểu thức này sẽ bị lỗi nếu **myNullableValue** là null.
* Sử dụng từ khóa **required** để bắt buộc một tham số không thể null trong hàm hoặc phương thức. Ví dụ: **MyClass({required this.myRequiredParam});**. Khi tạo một đối tượng của lớp **MyClass**, ta bắt buộc phải truyền một giá trị cho **myRequiredParam** và giá trị này không thể là null.
* Các quan hệ này có thể được kết hợp để định nghĩa các biến hoặc tham số có thể null hoặc không null. Ví dụ: **String? myNullableString;** để định nghĩa một biến có thể null kiểu **String**, hoặc **int myNonNullInt = myNullableInt ?? 0;** để gán giá trị mặc định là 0 cho biến **myNonNullInt** nếu **myNullableInt** là null.

### Chương 3: State

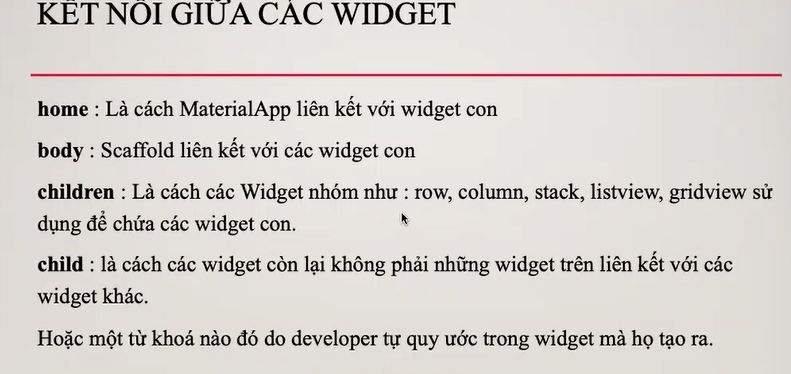
* Trạng thái của UI. State thay đổi thì UI thay đổi.Chỉ dùng trong màn hình class và sử dụng StatefulWidget.
* Các giai đoạn trong State:
* 1. creatState() : Hàm tạo ra state và tạo ra BuidContext gắn với State đó.
* 2. mounted = true : biến global kiểm tra xem State đã có chưa.Tồn tại mounted = true , ngược lại là false.
* 3. initSate(): không gọi nó vẫn chạy là 1 hàm mặc định.Khởi tạo State.
* 4. didChangDependencies(): gọi ngay sau innitState() .Hàm chạy khi lần đầu chạy widget, chỉ chạy sau khi build có sự thay đổi. Có thể bắt đầu dùng context.
* 5. Hàm build để build giao diện hiển thị ra màn hình người dùng: state, context,..

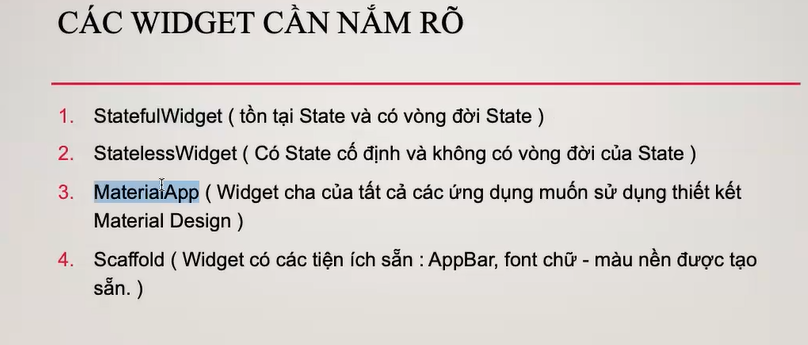
###### HỌC Ở LỚP

* Widget là 1 loại class được dùng để sử dụng cho việc sử dụng giao diện.
* Widget là View
* Có hai loại Widget:
* Stateless Widget không sử dụng State.
* Stateful widget gắn liền với State.
* Layout : text, grow, stack.
* Component: TextFile, button,..
* Hai chuẩn : material,..

#### WIDGET

* Hai chuẩn trong flutter là Material và Cupertino
* Widget là View( là nhưng cái trong giao diện).
* Tạo StatelessWidget : stl enter.
* Tạo StatefulWidget : stf enter.
* const : mặc định không thay đổi( để chương trình không phải biuld lại khi chạy lại)





* padding: const EdgeInsets.only(right : 30, left: 30, top: 30)
* Const SizeBox ( thuộc tính: đơn vị (height : 20 ->trên dưới 20 ) : Tạo khoảng cách
* Hàm reload: setState((){//code});
* Ngày tháng năm hiện tại: DateTime.Now(); DateTime.Now().year.