LẬP TRÌNH FLUTTER - Chuyển đổi ngôn ngữ

[Bài trước](https://hiepsiit.com/detail/flutter/lap-trinh-flutter/khai-niem-ve-database)

[Bài sau](https://hiepsiit.com/detail/flutter/lap-trinh-flutter/testing)

Ngày nay, các ứng dụng di động được sử dụng từ nhiều nơi trên thế giới, vì thế các ứng dụng phải hiển thị nội dung phù hợp với ngôn ngữ của quốc gia đó (ví dụ ngườ Pháp thì nội dung hiển thị là tiếng Pháp, người Việt thì hiển thị tiếng Việt).Việc ứng dụng làm việc với đa ngôn ngữ được gọi là Internationalizing(quốc tế hoá).

Để ứng dụng làm việc với nhiều ngôn ngữ, đầu tiên nên tìm ngôn ngữ hiện tại của hệ thống mà ứng dụng đang chạy và sau đó hiển thị nội dung ở vị trí cụ thể  và quy trình này được gọi là Localization

Flutter framework cung cấp 3 lớp localization và các lớp tiện ích có nguồn gốc từ các lớp dựa trên localize

Các lớp cơ sở như sau :

*Locale -*là lớp được sử dụng để nhận diện ngôn ngữ người sử dụng. Ví dụ  en-us  nhận biết người Mỹ, người Anh và nó được tạo ra như sau :

Locale en\_locale = Locale('en', 'US')

Ở đây, đối số đầu tiên là mã ngôn ngữ, đối số thứ hai là mã quốc gia. Một ví dụ khác và tạo *Argentina Spanish (es-ar)*như sau :

Locale es\_locale = Locale('es', 'AR')

*Localizations -*là widget chung được sử dụng để set Locale và nguồn localized của lớp con

class CustomLocalizations {

CustomLocalizations(this.locale);

final Locale locale;

static CustomLocalizations of(BuildContext context) {

return Localizations.of<CustomLocalizations>(context, CustomLocalizations);

}

static Map<String, Map<String, String>> \_resources = {

'en': {

'title': 'Demo',

'message': 'Hello World'

},

'es': {

'title': 'Manifestación',

'message': 'Hola Mundo',

},

};

String get title {

return \_resources[locale.languageCode]['title'];

}

String get message {

return \_resources[locale.languageCode]['message'];

}

}

Ở đây, CustomLocalizations là lớp custom mới được tạo riêng để lấy nội dung localized nhất định (tiêu đề và thông báo) cho widget,of của phương thức sử dụng lớp Localizations để trả về lớp CustomLocalizations mới

LocalizationsDelegate<T> - LocalizationsDelegate<T> là lớp factory thông qua widget Localizations được tải. Nó có 3 phương thức over-ridable như sau :

\* sSupported - Chấp nhận một miền - và trả về liệu miền đó có được hỗ trợ hay không?

@override

bool isSupported(Locale locale) => ['en', 'es'].contains(locale.languageCode);

Ở đây chỉ làm việc với *en* hoặc *es*

\* load - Chấp nhận ngôn ngữ được chọn và bắt đầu tải các nguồn dữ liệu của ngôn ngữ đó

@override

Future<CustomLocalizations> load(Locale locale) {

return SynchronousFuture<CustomLocalizations>(CustomLocalizations(locale));

}

Ở trên, phương thức load trả về CustomLocalizations. việc trả về CustomLocalizations có thể được sử dụng để lấy giá trị của tiêu đề và thông báo của cả 2 ngôn ngữ Englist và Spanish

\* shouldReload - Liệu có nên tải lại CustomLocalizations là cần thiết khi widget Localizations được rebuild(giống như việc reset lại trang)

@override

bool shouldReload(CustomLocalizationsDelegate old) => false;

Code ở CustomLocalizationDelegate như sau :

class CustomLocalizationsDelegate extends

LocalizationsDelegate<CustomLocalizations> {

const CustomLocalizationsDelegate();

@override

bool isSupported(Locale locale) => ['en', 'es'].contains(locale.languageCode);

@override

Future<CustomLocalizations> load(Locale locale) {

return SynchronousFuture<CustomLocalizations>(CustomLocalizations(locale));

}

@override bool shouldReload(CustomLocalizationsDelegate old) => false;

}

Nhìn chung, ứng dụng Flutter dựa trên 2 cấp widget gốc là MaterialApp và WidgetsApp. Flutter cung cấp miền cho cả 2 widget và nó là MaterialLocalizations và WidgetsLocaliations . Thêm nữa , flutter cũng cung cấp quyền để tải MaterialLocalizations và WidgetsLocaliations, đó là GlobalMaterialLocalizations.delegate và GlobalWidgetsLocalizations.delegate tương ứng. Chúng ta hãy tạo ứng dụng đa ngôn ngữ cơ bản để chạy thử và hiểu về nội dung

Đầu tiên, ta tạo một ứng dụng mới với tên  *flutter\_localization\_app*

Flutter hỗ trợ đa ngôn ngữ(internationalization) sử dụng gói flutter là *flutter\_localizations*. Mở file pubspec.yaml và thêm vào như sau :

dependencies:

flutter:

sdk: flutter

flutter\_localizations:

sdk: flutter

Sau đó ta Get dependencies khi IDE hiện thông báo cập nhật

Thêm flutter\_localizations vào main.dart:

import 'package:flutter\_localizations/flutter\_localizations.dart';

import 'package:flutter/foundation.dart' show SynchronousFuture;

Ở đây, mục đích của SynchronousFuture để tải tuỳ chỉnh localizations không đồng bộ

Tạo custom localizations như sau :

class CustomLocalizations {

CustomLocalizations(this.locale);

final Locale locale;

static CustomLocalizations of(BuildContext context) {

return Localizations.of<CustomLocalizations>(context, CustomLocalizations);

}

static Map<String, Map<String, String>> \_resources = {

'en': {

'title': 'Demo',

'message': 'Hello World'

},

'es': {

'title': 'Manifestación',

'message': 'Hola Mundo',

},

};

String get title {

return \_resources[locale.languageCode]['title'];

}

String get message {

return \_resources[locale.languageCode]['message'];

}

}

class CustomLocalizationsDelegate extends

LocalizationsDelegate<CustomLocalizations> {

const CustomLocalizationsDelegate();

@override

bool isSupported(Locale locale) => ['en', 'es'].contains(locale.languageCode);

@override

Future<CustomLocalizations> load(Locale locale) {

return SynchronousFuture<CustomLocalizations>(CustomLocalizations(locale));

}

@override bool shouldReload(CustomLocalizationsDelegate old) => false;

}

Ở trên, CustomLocalizations được tạo để hỗ trợ miền cho tiêu đề và thông báo trong ứng dụng và  CustomLocalizationsDelegate được sử dụng để tải CustomLocalizations

Thêm quyền cho MaterialApp, WidgetsApp và CustomLocalization sử dụng thuộc tính MaterialApp, localizationsDelegates và supportedLocales như sau :

localizationsDelegates: [

const CustomLocalizationsDelegate(),

GlobalMaterialLocalizations.delegate,

GlobalWidgetsLocalizations.delegate,

],

supportedLocales: [

const Locale('en', ''),

const Locale('es', ''),

],

Sử dụng phương thức CustomLocalizations , *of*để lấy giá trị của localized và sử dụng nó thích hợp như sau :

class MyHomePage extends StatelessWidget {

MyHomePage({Key key, this.title}) : super(key: key);

final String title;

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text(CustomLocalizations .of(context) .title), ),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text( CustomLocalizations .of(context) .message, ),

],

),

),

);

}

}

Nhìn trên, ta đã sửa đổi lớp MyHomePage từ StatefulWidget sang StatelessWidget và sử dụng CustomLocalizations để lấy tiêu đề, thông điệp

Biên dịch và chạy ứng dụng . Ứng dụng sẽ hiển thị nội dung là tiếng Anh.

Đóng ứng dụng. Vào **Settings → System → Languages and Input → Languages\***

Nhấn thêm ngôn ngữ và lựa chọn Spanish. Điện thoại sẽ cài đặt ngôn ngữ Spanish

Lựa chọn Spanish và di chuyển nó lên trên English. Nó sẽ mặc định chọn ngôn ngữ Spanish là ngôn ngữ đầu tiên và tất cả sẽ được chuyển sang ngôn ngữ Spanish

Chúng ta có thể thay đổi lại ngôn ngữ tiến Anh bằng cách tương tự là di chuyển English lên trên đầu trong cài đặt

Và kết quả hiển thị như sau :



Sử dụng gói intl

Flutter cung cấp intl package để đơn giản việc phát triển localized trong ứng dụng mobile.intl package cung cấp phương thức đặc biệt và công cụ để tạo bán tự động ngôn ngữ thông điệp cụ thể

Chúng ta sẽ tạo ứng dụng localized mới bằng việc sử dụng intl package và hiểu về ý tưởng của package này

- Tạo ứng dụng flutter mới với tên *"flutter\_intl\_app"*

- Mở file  pubspec.yaml và thêm package như sau :

dependencies:

flutter:

sdk: flutter

flutter\_localizations:

sdk: flutter

intl: ^0.15.7

intl\_translation: ^0.17.3

Get dependencies khi Android studio hiển thị thông báo câp nhật thay đổi

- Copy hàm main.dart trong ví dụ trước *flutter\_internationalization\_app*

- Import intl package như sau

import 'package:intl/intl.dart';

Cập nhật lớp CustomLocalization :

class CustomLocalizations {

static Future<CustomLocalizations> load(Locale locale) {

final String name = locale.countryCode.isEmpty ? locale.languageCode : locale.toString();

final String localeName = Intl.canonicalizedLocale(name);

return initializeMessages(localeName).then((\_) {

Intl.defaultLocale = localeName;

return CustomLocalizations();

});

}

static CustomLocalizations of(BuildContext context) {

return Localizations.of<CustomLocalizations>(context, CustomLocalizations);

}

String get title {

return Intl.message(

'Demo',

name: 'title',

desc: 'Title for the Demo application',

);

}

String get message{

return Intl.message(

'Hello World',

name: 'message',

desc: 'Message for the Demo application',

);

}

}

class CustomLocalizationsDelegate extends

LocalizationsDelegate<CustomLocalizations> {

const CustomLocalizationsDelegate();

@override

bool isSupported(Locale locale) => ['en', 'es'].contains(locale.languageCode);

@override

Future<CustomLocalizations> load(Locale locale) {

return CustomLocalizations.load(locale);

}

@override

bool shouldReload(CustomLocalizationsDelegate old) => false;

}

Ở trên, ta đã sử dụng 3 phương thức từ intl package thay vì phương thức tự tạo .

1. Intl.canonicalizedLocale - được sử dụng để lấy chính xác tên ngôn ngữ
2. Intl.defaultLocale  - Sử dụng để set ngôn ngữ hiện tại
3. Intl.message - Sử dụng để định nghĩa thông điệp mới

import **l10n/messages\_all.dart** file

import 'l10n/messages\_all.dart';

Tiếp theo, ta tạo thư mục lib/l10n

Mở command prompt và đi đến ứng dụng thư mục gốc (pubspec.yaml) và chạy đoạn command sau :

flutter packages pub run intl\_translation:extract\_to\_arb --output-

dir=lib/l10n lib/main.dart

Ở đây, lệnh được tạo intl\_message.arb file, Một bản mẫu để tạo thông điệp trong ngôn ngữ khác . Nội dung của file như sau :

{

"@@last\_modified": "2019-04-19T02:04:09.627551",

"title": "Demo",

"@title": {

"description": "Title for the Demo application",

"type": "text",

"placeholders": {}

},

"message": "Hello World",

"@message": {

"description": "Message for the Demo

application",

"type": "text",

"placeholders": {}

}

}

Copy intl\_message.arb và tạo file mới, intl\_en.arb và thay đổi nội dung sang ngôn ngữ Spanish :

{

"@@last\_modified": "2019-04-19T02:04:09.627551",

"title": "Manifestación",

"@title": {

"description": "Title for the Demo application",

"type": "text",

"placeholders": {}

},

"message": "Hola Mundo",

"@message": {

"description": "Message for the Demo application",

"type": "text",

"placeholders": {}

}

}

Oke, bây giờ ta thử run lệnh để tạo tệp cuối cùng, messages\_all.dart.

flutter packages pub run intl\_translation:generate\_from\_arb

--output-dir=lib\l10n --no-use-deferred-loading

lib\main.dart lib\l10n\intl\_en.arb lib\l10n\intl\_es.arb

Biên dịch và run app .Nó sẽ hoạt động như ứng dụng flutter\_localization\_app ở trên .