

**TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA ĐIỆN TỬ VIỄN THÔNG**

**BÁO CÁO  
LẬP TRÌNH ĐA NỀN TẦNG**

**Chủ đề: InheritedWidget trong Flutter**

**Sinh viên thực hiện:**

**01. Đặng Quang Lâm** Lớp: 22KTMT2 MSSV: 106220256  
**02. Nguyễn Hữu Nam** Lớp: 22KTMT2 MSSV: 106220262

**Người hướng dẫn:**

**TS. Nguyễn Duy Nhật Viễn**

**Đà Nẵng, 2025.**

## THUYẾT MINH

## BÁO CÁO

# LẬP TRÌNH ĐA NỀN TẢNG

## Chủ đề: InheritedWidget trong Flutter

### BẢNG PHÂN CÔNG CÔNG VIỆC TRONG NHÓM

STT	HỌ VÀ TÊN	NHIỆM VỤ	KHỐI LUỢNG
01	Đặng Quang Lâm	Tạo custom InheritedWidget và Context.dependOnInheritedWidgetOfExactType()	50%
02	Nguyễn Hữu Nam	Performance considerations và khi nào nên sử dụng InheritedWidget	50%

Link code github: [github](#)

## Mục lục

<b>1. Giới thiệu về InheritedWidget.....</b>	<b>4</b>
<b>1.1 Khái niệm .....</b>	<b>4</b>
<b>1.2 Vai trò của InheritedWidget.....</b>	<b>4</b>
<b>2. Cấu trúc và các sử dụng InheritedWidget trong flutter .....</b>	<b>5</b>
<b>2.1 Cấu trúc của InheritedWidget .....</b>	<b>5</b>
<b>2.2 Tạo Custom InheritedWidget .....</b>	<b>6</b>
<b>2.3. Sử dụng context.dependOnInheritedWidgetOfExactType()</b> .....	<b>7</b>
<b>2.4 InheritedWidget Demo .....</b>	<b>7</b>
<b>3. Các điểm cần chú ý về hiệu năng (Performance Considerations): .....</b>	<b>10</b>
<b>3.1. Rebuild không cần thiết: .....</b>	<b>10</b>
<b>3.2. Phạm vi ảnh hưởng quá rộng: .....</b>	<b>10</b>
<b>3.3. Tách nhỏ widget để giảm cập nhật:.....</b>	<b>10</b>
<b>3.4. Ưu tiên dữ liệu bất biến:.....</b>	<b>10</b>
<b>3.5. Không nên lạm dụng: .....</b>	<b>11</b>
<b>3.6. InheritedModel tối ưu hơn: .....</b>	<b>11</b>
<b>4. Khi nào nên dùng InheritedWidget:.....</b>	<b>11</b>

## 1. Giới thiệu về InheritedWidget

Trong lập trình Flutter, quản lý trạng thái là một trong những yếu tố quan trọng để xây dựng ứng dụng hiệu quả, dễ bảo trì. Một trong những cơ chế cốt lõi mà Flutter cung cấp để chia sẻ dữ liệu giữa các widget là InheritedWidget. Đây là nền tảng cho các thư viện quản lý trạng thái phổ biến như Provider, Riverpod...

### 1.1 Khái niệm

InheritedWidget là một lớp trừu tượng trong Flutter, cho phép truyền dữ liệu từ widget cha xuống các widget con trong cây widget mà không cần truyền qua từng lớp trung gian.

Đặc điểm:

- Được sử dụng để chia sẻ dữ liệu trong cây widget.
- Các widget con có thể “lắng nghe” và tự động rebuild khi dữ liệu thay đổi.
- Là một phần của hệ thống widget của Flutter, không cần cài thêm thư viện.
- Các widget con có thể truy cập trực tiếp dữ liệu bằng context.

Ví dụ:

```
final theme = Theme.of(context);
final media = MediaQuery.of(context);
```

### 1.2 Vai trò của InheritedWidget

- Giúp truyền dữ liệu xuyên suốt cây widget
- Truyền dữ liệu hiệu quả, giảm việc truyền thủ công qua nhiều lớp (prop drilling)
- Là cơ sở cho Provider, Riverpod, Bloc...
- Tối ưu hiệu năng khi dữ liệu thay đổi

Ví dụ:

```
class AppInfo extends InheritedWidget {  
    final String appName = "Flutter Demo App";  
    const AppInfo({required super.child});  
    static AppInfo of(BuildContext context) =>  
        context.dependOnInheritedWidgetOfExactType<AppInfo>  
    ()!;  
    @override  
    bool updateShouldNotify(AppInfo oldWidget) => false;  
}  
  
class DeepChild extends StatelessWidget {  
    @override  
    Widget build(BuildContext context) {  
        final info = AppInfo.of(context);  
        return Text("Ứng dụng: ${info.appName}");  
    }  
}
```

## 2. Cấu trúc và các sử dụng InheritedWidget trong flutter

### 2.1 Cấu trúc của InheritedWidget

- Dữ liệu muôn chia sẻ
- Hàm static of(context) để truy cập
- updateShouldNotify() để kiểm tra khi nào rebuild

Ví dụ:

```

class MyData extends InheritedWidget {
    final String message;
    const MyData({required this.message, required
super.child});
    static MyData of(BuildContext context) =>
        context.dependOnInheritedWidgetOfExactType<MyData>()
!;
    @override
    bool updateShouldNotify(MyData oldWidget) =>
        message != oldWidget.message;
}

```

## 2.2 Tạo Custom InheritedWidget

- Tạo lớp kế thừa InheritedWidget
- Thêm dữ liệu cần chia sẻ
- Dùng of(context) để truy cập

Ví dụ:

```

class MyData extends InheritedWidget {
    final int counter;
    const MyData({required this.counter, required super.child});
    static MyData of(BuildContext context) =>
        context.dependOnInheritedWidgetOfExactType<MyData>()!;
    @override
    bool updateShouldNotify(MyData old) => counter != old.counter;
}

```

### 2.3. Sử dụng context.dependOnInheritedWidgetOfExactType()

Đây là phương thức được dùng để truy cập dữ liệu từ InheritedWidget.

Cách hoạt động:

- Tìm InheritedWidget tương ứng trong cây widget
- Đăng ký widget hiện tại để rebuild khi InheritedWidget thay đổi

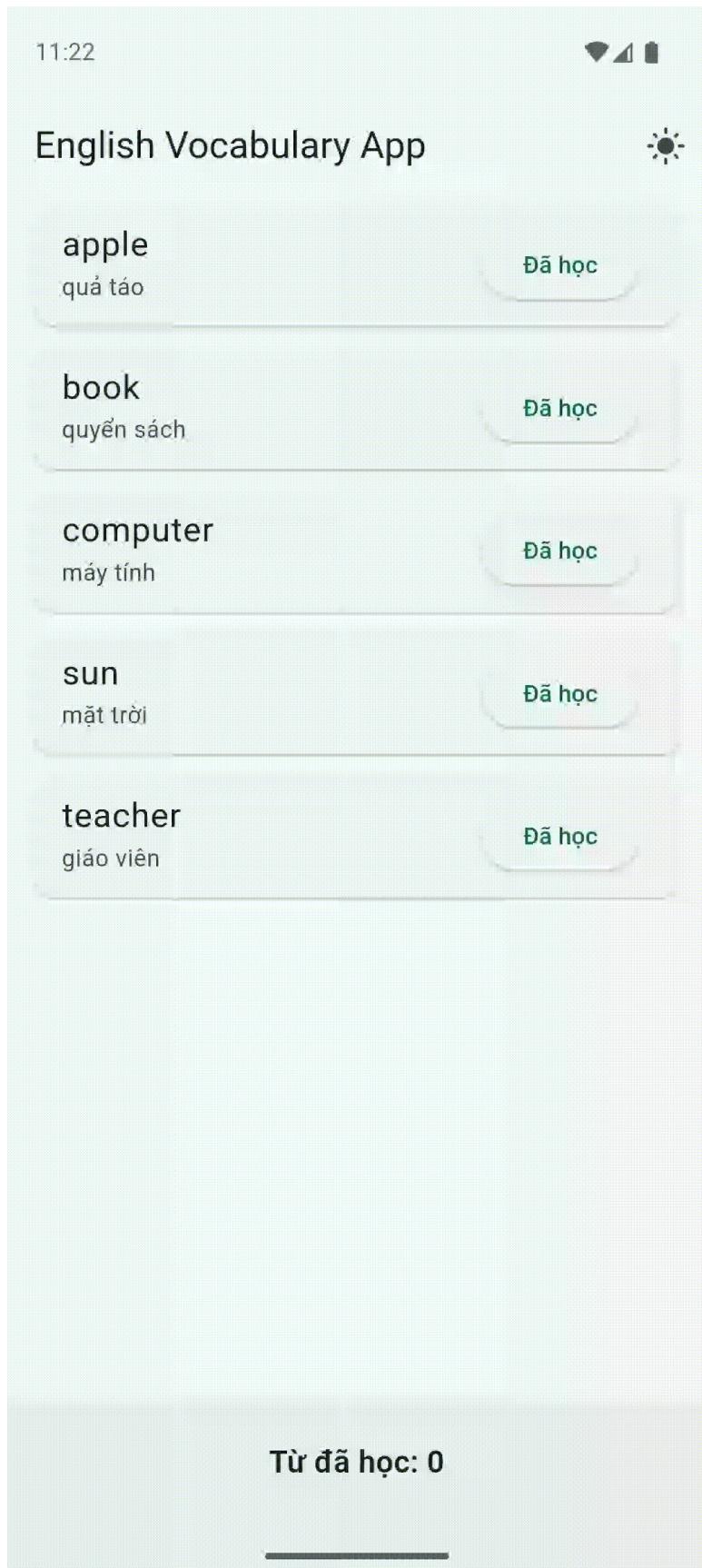
Ví dụ:

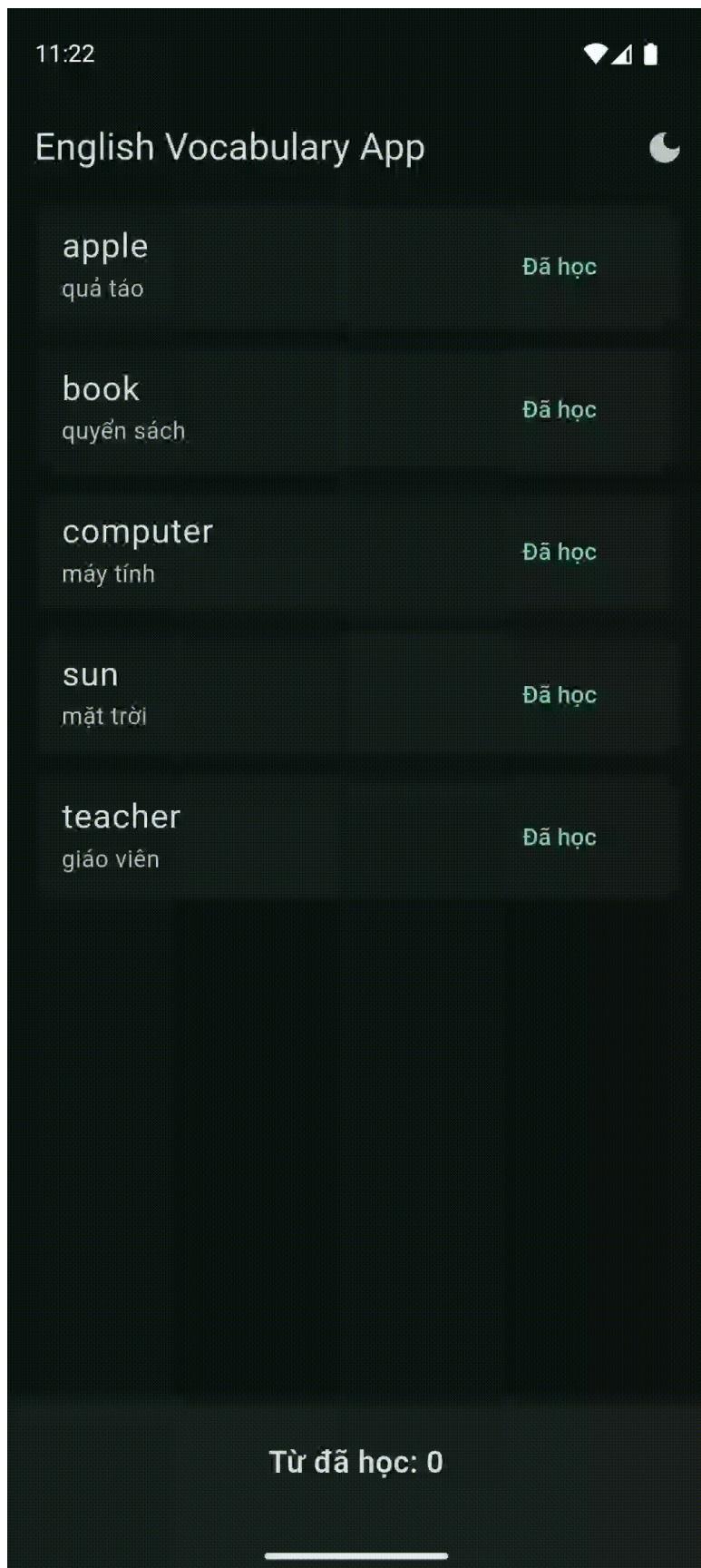
```
class CounterInheritedWidget extends InheritedWidget {
    final int counter;
    const CounterInheritedWidget({
        required this.counter,
        required super.child,
    });
    static CounterInheritedWidget of(BuildContext context) =>
        context.dependOnInheritedWidgetOfExactType<CounterInheritedWidget>()!;
    @override
    bool updateShouldNotify(CounterInheritedWidget oldWidget) =>
        oldWidget.counter != counter;
}
```

### 2.4 InheritedWidget Demo

Ứng dụng gồm 3 phần chính:

- InheritedWidget quản lý trạng thái toàn app (theme + từ vựng đã học)
- Danh sách từ vựng để học
- Nút đổi giao diện Dark/Light mode





### 3. Các điểm cần chú ý về hiệu năng (Performance Considerations):

Khi dùng InheritedWidget, cần quan tâm cách nó ảnh hưởng đến quá trình rebuild trong Flutter để tránh ứng dụng chạy chậm, tốn tài nguyên, hoặc render không cần thiết... Từ đó dẫn đến ảnh hưởng về mặt hiệu năng (app chậm/lag)

#### 3.1. Rebuild không cần thiết:

- Khi dữ liệu trong InheritedWidget thay đổi  
=> tất cả widget đang phụ thuộc (dependOn) sẽ bị rebuild.
- Nếu nhiều widget phụ thuộc  
=> rebuild nhiều lần => tốn CPU.
- Không kiểm soát tốt  
=> hiệu năng giảm.

Ví dụ: `context.dependOnInheritedWidgetOfExactType()`  
=> Widget rebuild lại khi InheritedWidget cập nhật

#### 3.2. Phạm vi ảnh hưởng quá rộng:

- Nên hạn chế phạm vi ảnh hưởng của InheritedWidget, nếu đặt quá cao trong widget tree  
=> nhiều widget sẽ chịu ảnh hưởng => dẫn đến rebuild lan rộng.
- Đặt InheritedWidget gần nơi cần dùng để giới hạn phạm vi rebuild.

#### 3.3. Tách nhỏ widget để giảm cập nhật:

- Widget build phức tạp gây tốn thời gian  
=> nên tách thành widget nhỏ.
- Mục tiêu: mỗi thay đổi chỉ ảnh hưởng phạm vi nhỏ.
- “Smaller widgets → less rebuild → faster”

#### 3.4. Ưu tiên dữ liệu bất biến:

- Nên dùng dữ liệu bất biến vì Flutter dễ nhận biết thay đổi giữa object cũ và mới  
=> rebuild đúng widget cần thiết => hiệu năng tốt hơn.
- Khi cập nhật dữ liệu  
=> tạo object mới thay vì sửa object cũ
- Sử dụng copyWith() để sao chép và thay đổi một phần thuộc tính
- Tránh cập nhật trực tiếp giá trị bên trong object cũ tránh UI rebuild sai hoặc không rebuild

### 3.5. Không nên lạm dụng:

- InheritedWidget phù hợp cho state nhỏ, thay đổi ít
- State thay đổi liên tục  
=> rebuild nhiều => giảm hiệu năng, app kém mượt
- State quá phức tạp  
=> nên cân nhắc sử dụng thư viện khác Provider / Riverpod / ...

### 3.6. InheritedModel tối ưu hơn:

- InheritedWidget  
=> dù chỉ 1 phần dữ liệu thay đổi => rebuild tất cả widget phụ thuộc
- InheritedModel  
=> khi 1 phần dữ liệu thay đổi => chỉ widget “quan tâm” phần đó mới rebuild
- Phù hợp khi nhiều widget dùng các phần khác nhau từ cùng một dữ liệu

## 4. Khi nào nên dùng InheritedWidget:

- InheritedWidget phù hợp cho những trường hợp truyền dữ liệu xuống sâu trong cây widget mà không cần cập nhật thường xuyên
- Dữ liệu thay đổi ít hoặc hiếm khi thay đổi
- Muốn sử dụng thuận Flutter SDK, không muốn dùng package ngoài
- Phù hợp state đơn giản, ít lớp, ít logic cập nhật
- Xây dựng state management cho mục đích học tập / học thuật / tìm hiểu cơ chế quản lý state ở mức thấp

Ví dụ: + *App theme*  
 + *Language*  
 + *Settings*  
     + *Thông tin đăng nhập cơ bản*  
 + ...

## Vậy thì khi nào không nên dùng InheritedWidget?

- State thay đổi liên tục
- State độ phức tạp cao, nhiều logic
- Nhiều widget dùng các phần dữ liệu khác nhau
- App/Dự án quy mô lớn, gồm nhiều lớp
- Không phù hợp nếu cần quản lý UI state

Ví dụ: + *Real-time stream*  
 + *Socket*

+ State phúc tạp

## TÀI LIỆU THAM KHẢO

- [1]. Flutter, “InheritedWidget class — widgets library — Dart API,” Available: <https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html>. Accessed on: Nov. 11, 2025.
- [2]. Flutter, “State management — Flutter Docs,” Available: <https://docs.flutter.dev/data-and-backend/state-mgmt/options>. Accessed on: Nov. 11, 2025.
- [3]. Flutter, “Fundamentals: State management,” Available: <https://docs.flutter.dev/get-started/fundamentals/state-management>. Accessed on: Nov. 11, 2025.
- [4]. N. T. Minh, “Học Flutter từ cơ bản đến nâng cao. Phần 4: Lột trần InheritedWidget,” Viblo, Sep. 06, 2020. Available: <https://viblo.asia/p/hoc-flutter-lot-tran-inheritedwidget>

tu-co-ban-den-nang-cao-phan-4-lot-tran-inheritedwidget-3P0lPDbmlox. Accessed on: Nov. 11, 2025.