



02-08 flutter InheritedWidget

Posted on 2020-11-23 23:41 肖无情 阅读(100) 评论(0) 编辑 收藏 举报

InheritedWidget提供了一种数据在widget树中从上到下传递、共享的方式，

简而言之

InheritedWidget 中暴露出来的数据能有效地向下(子widget)传播 (和共享) 信息

如Flutter SDK中正是通过InheritedWidget来共享应用主题 (Theme) 和Locale (当前语言环境)信息的。

InheritedWidget

```
abstract class InheritedWidget extends ProxyWidget
```

为了使用 先定义一个子类InheritedProvider

```
class InheritedProvider<T> extends InheritedWidget {

    final T data; //需要在子树中共享的数据

    InheritedProvider({
        this.data,
        Widget child,
    }) :super(child: child);

    //定义一个便捷方法, 方便子树中的widget获取共享数据
    static InheritedProvider of<T>(BuildContext context) {
        return context.inheritFromWidgetOfExactType(InheritedProvider);
    }
    //该回调决定当data发生变化时, 是否通知子树中依赖data的Widget
    @override
    bool updateShouldNotify(InheritedProvider<T> oldWidget) {
        //如果返回true, 则子树中依赖(build函数中有调用)本widget
        //的子widget的`state.didChangeDependencies`会被调用
        return oldWidget.data != data;
    }
}
```

InheritedProvider类的目的是为它的所有子widget提供数据

InheritedWidget在widget树中数据传递方向是从上到下的，并且可以跨级传递

访问inhertedProvider中的数据

```
final InheritedProvider inheritedWidget = InheritedProvider.of(context);

return new Container(
    color: inheritedWidget.data.color,
);
```

didChangeDependencies

State对象有一个didChangeDependencies回调，它会在“依赖”发生变化时被Flutter Framework调用。而这个“依赖”指的就是子widget是否使用了父widget中InheritedWidget的数据！如果使用了，则代表子widget依赖有依赖InheritedWidget；如果没有使用则代表没有依赖。这种机制可以使子组件在所依赖的InheritedWidget变化时来更新自身！比如当主题、locale(语言)等发生变化时，依赖其的子widget的didChangeDependencies方法将会被调用。

如果我们只想在子widget中引用InheritedProvider数据，但却不希望InheritedProvider发生变化时调用子Widget的didChangeDependencies()方法应该怎么办？其实答案很简单，我们只需要将InheritedProvider.of()的实现改一下即可：

```
//定义一个便捷方法, 方便子树中的widget获取共享数据
static InheritedProvider of(BuildContext context) {
    //return context.dependOnInheritedWidgetOfExactType<ShareDataWidget>();
    return context.getElementForInheritedWidgetOfExactType<ShareDataWidget>().widget;
}
```

```
@override
InheritedElement getElementForInheritedWidgetOfExactType<T extends InheritedWidget>() {
    assert(_debugCheckStateIsActiveForAncestorLookup());
    final InheritedElement ancestor = _inheritedWidgets == null ? null : _inheritedWidgets[T];
    return ancestor;
}

@override
InheritedWidget dependOnInheritedWidgetOfExactType({ Object aspect }) {
    assert(_debugCheckStateIsActiveForAncestorLookup());
    final InheritedElement ancestor = _inheritedWidgets == null ? null : _inheritedWidgets[T];
    //多出的部分
    if (ancestor != null) {
        assert(ancestor is InheritedElement);
        return dependOnInheritedElement(ancestor, aspect: aspect) as T;
    }
    _hadUnsatisfiedDependencies = true;
    return null;
}
```

我们可以看到，dependOnInheritedWidgetOfExactType() 比 getElementForInheritedWidgetOfExactType()多调了dependOnInheritedElement方法，dependOnInheritedElement源码如下：

```
@override
InheritedWidget dependOnInheritedElement(InheritedElement ancestor, { Object aspect }) {
    assert(ancestor != null);
```

```

    _dependencies ??= HashSet<InheritedElement>();
    _dependencies.add(ancestor);
    ancestor.updateDependencies(this, aspect);
    return ancestor.widget;
}

```

调用`dependOnInheritedWidgetOfExactType()` 和 `getElementForInheritedWidgetOfExactType()`的区别就是前者会注册依赖关系，而后者不会，所以在调用`dependOnInheritedWidgetOfExactType()`时，`InheritedWidget`和依赖它的子孙组件关系便完成了注册，之后当`InheritedWidget`发生变化时，就会更新依赖它的子孙组件，也就是会调这些子孙组件的`didChangeDependencies()`方法和`build()`方法。而当调用的是 `getElementForInheritedWidgetOfExactType()`时，由于没有注册依赖关系，所以之后当`InheritedWidget`发生变化时，就不会更新相应的子孙`Widget`。

点击"Increment"按钮后，会调用界面`state`的`setState()`方法，此时会重新构建整个页面，由于示例中，并没有任何缓存，所以它也都会被重新构建，所以也会调用`build()`方法。

分类: flutter

标签: flutter

好文要顶

关注我

收藏该文





肖无情

粉丝 - 2 关注 - 5

+ 加关注


0 推荐

0 反对

« 上一篇: 02-13 flutter+mvp

» 下一篇: 02-09 flutter状态保持

[刷新评论](#) [刷新页面](#) [返回顶部](#)

 登录后才能查看或发表评论，立即 登录 或者 逛逛 博客园首页

【推荐】阿里云新人特惠，爆款云服务器2核4G低至0.46元/天

编辑推荐:

- gRPC 入门与实操 (.NET 篇)
- dotnet 代码优化 聊聊逻辑圈复杂度
- 一个棘手的生产问题，但是我写出来之后，就是你的了
- 你可能不知道的容器镜像安全实践
- .Net 6 使用 Consul 实现服务注册与发现

阅读排行:

- Redux与前端表格施展“组合拳”，实现大屏展示应用的交互增强
- gRPC入门与实操(.NET篇)
- 博客园主题修改分享 - 过年篇
- 如何优雅地校验后端接口数据，不做前端背锅侠
- 产品与研发相处之道