

Flutter - Ресурсы приложения

Flutter-проект может содержать не только файлы с кодом, но и различные файлы ресурсов, которые называются ассетами (англ. assets). К самым распространённым типам ресурсов относятся статические данные, файлы конфигурации, иконки и изображения.

Как добавить новый ресурс

- Для начала нам необходимо перенести нужный файл в папку проекта. Общепринятое решение — создавать папку с именем [assets](#) на уровне папки [lib](#) и хранить все ресурсы проекта в ней. Дальше необходимо указать расположение ресурса в файле [pubspec.yaml](#) в секции [flutter](#):

Изображения

- Пример, как добавить изображение в файл проекта pubspec.yaml:

```
flutter:  
  assets:  
    - assets/folder_name/asset_name.png ## Добавится изображение png  
    - assets/folder_name/asset_name.svg ## Добавится изображение svg  
    - assets/folder_name/ ## Добавятся все ресурсы из папки assets/folder_name  
    - my_folder/ ## Добавятся все ресурсы из папки my_folder  
    - packages/package_name/assets/folder_name/asset_name.png ## Добавится asset_name.png
```

После добавления в проект доступ к изображению можно получить через класс [AssetImage](#) или [Image.asset](#). Например, для показа изображения, которое находится по пути [assets/folder name/asset name.png](#), нужно создать экземпляр [Image.asset](#) таким образом:

```
Image.asset\('assets/folder name/asset name.png'\).
```

Зависимые от pixel ratio изображения

- Flutter может загружать изображения с соответствующим разрешением для текущего соотношения пикселей устройства ([англ. pixel ratio](#)).

Что такое pixel ratio

- Это отношение между физическим количеством пикселей устройства и логическим количеством пикселей, с которым работает приложение.
- В мобильных устройствах используется различное физическое разрешение экрана. Устройства с более высоким разрешением имеют большее количество физических пикселей на экране, что позволяет отображать изображения с более высокой чёткостью и детализацией.
- При этом в разработке приложений часто используется логическое разрешение, которое измеряется в логических пикселях.
- Логический пиксель — абстрактная единица измерения, используемая для создания интерфейса приложения.
- Коэффициент пикселей устройства позволяет адаптировать приложение к физическому разрешению экрана устройства. Он определяет, какое количество физических пикселей будет использоваться для отображения каждого логического пикселя в приложении. Если устройство имеет коэффициент пикселей устройства, отличный от 1.0, то для каждого логического пикселя можно использовать несколько физических пикселей для отображения более чёткого и детализированного изображения.
- Например, устройство с pixel ratio 2.0 будет использовать четыре физических пикселя (два по горизонтали и два по вертикали) для отображения каждого логического пикселя.

- Класс [AssetImage](#) будет автоматически сопоставлять запрошенное изображение с наиболее подходящим по соотношению пикселей устройства.
- Для работы автоматического сопоставления ресурсы должны быть организованы согласно определённой структуре каталогов:

```
my_icon/my_icon.png      (mdpi)
my_icon/1.5x/my_icon.png (hdpi)
my_icon/2.0x/my_icon.png (xhdpi)
my_icon/3.0x/my_icon.png (xxhdpi)
my_icon/4.0x/my_icon.png (xxxhdpi)
```

```
icons/icon.png
icons/Mx/icon.png
icons/Nx/icon.png
```

- M и N — числовые идентификаторы, соответствующие разрешению изображения. Они указывают на соотношение пикселей устройства, для которого предназначены изображения. В этом примере [icon.png](#) считается основным ресурсом, а Mx/icon.png и Nx/icon.png — вариантами. Предполагается, что основной ресурс соответствует разрешению [1.0](#).
- Пример, как структурировать ресурсы для изображения с именем [my_icon.png](#):

- На устройствах с коэффициентом пикселей 1.8 выбирается ресурс 2.0x/my_icon.png. Для устройства с коэффициентом пикселей 2.7 выбирается ресурс 3.0x/my_icon.png.
- Если ширина и высота отображаемого изображения не указаны в виджете icon, используется номинальное разрешение для масштабирования ресурса таким образом, чтобы он занимал такое же пространство на экране, как основной ресурс, но с более высоким разрешением. Другими словами, если my_icon.png имеет размер 72 пикселя по ширине и 72 пикселя по высоте, то 3.0x/my_icon.png должно иметь размер 216 пикселей по ширине и 216 пикселей по высоте. Однако, если ширина и высота не указаны, оба ресурса будут отображаться как 72 пикселя по ширине и 72 пикселя по высоте (в логических пикселях).
- Для работы с ресурсами, зависимыми от разрешения, нужно указывать только основной ресурс или его родительский каталог. В примере ниже показаны оба варианта для my_icon:

```
flutter:  
  assets:  
    - my_icon/my_icon.png # Основной ресурс  
    - my_icon/ # Родительский каталог
```

Использование изображений из зависимостей

- Если изображение находится в импортируемом пакете, то для его использования необходимо указать поле [package](#) в `AssetImage`.
- Например, если пакет [my_icons](#) имеет такую структуру —

```
pubspec.yaml  
icons/heart.png  
icons/1.5x/heart.png  
icons/2.0x/heart.png
```

то загрузить изображение можно следующим образом:

```
AssetImage('icons/heart.png', package: 'my_icons')
```

Шрифты

- Для добавления шрифтов следует добавить файлы формата .ttc, .ttf или .otf, содержащие конфигурацию шрифта, и указать их в поле fonts:

```
flutter:  
  fonts:  
    - family: Raleway  
      fonts:  
        - asset: assets/fonts/Raleway-Regular.ttf  
        - asset: assets/fonts/Raleway-Italic.ttf  
          style: italic  
    - family: RobotoMono  
      fonts:  
        - asset: assets/fonts/RobotoMono-Regular.ttf  
        - asset: assets/fonts/RobotoMono-Bold.ttf  
          weight: 700
```

- Параметр `family` определяет семейство шрифтов. Этот параметр используется в свойстве `fontFamily` класса `TextStyle`.
- `asset` — путь к шрифту относительно файла `pubspec.yaml`. В примере `Raleway-Regular.ttf` лежит в папке `assets/fonts`, которая находится на том же уровне, что и `pubspec.yaml`.

Один шрифт может ссылаться на несколько файлов с различными стилями шрифтов:

- `weight` определяет насыщенность начертания, представляет собой числовое значение в диапазоне от `100` до `900`, кратное `100`. Это значение соответствует `FontWeight` и используется в свойстве `fontWeight` класса `TextStyle`;
- `style` определяет контур шрифта — курсивный или обычный. Значение соответствует полю `fontStyle` класса `TextStyle` и определяется объектом `FontStyle`. К примеру, для использования `Raleway-Italic` нужно установить `fontStyle` равным `FontStyle.italic`.

Текст

- Для работы с текстовыми [ассетами](#) — к примеру, файлами [.txt](#) или [.json](#) — используется [AssetBundle](#).

Есть два варианта работы с [AssetBundle](#):

Использование статического объекта rootBundle из package:flutter/services.dart

```
import 'package:flutter/services.dart' show rootBundle;

await rootBundle.loadString('assets/config.json');
```

Использование DefaultAssetBundle

```
import 'package:flutter/services.dart' show rootBundle;
import 'package:flutter/material.dart';

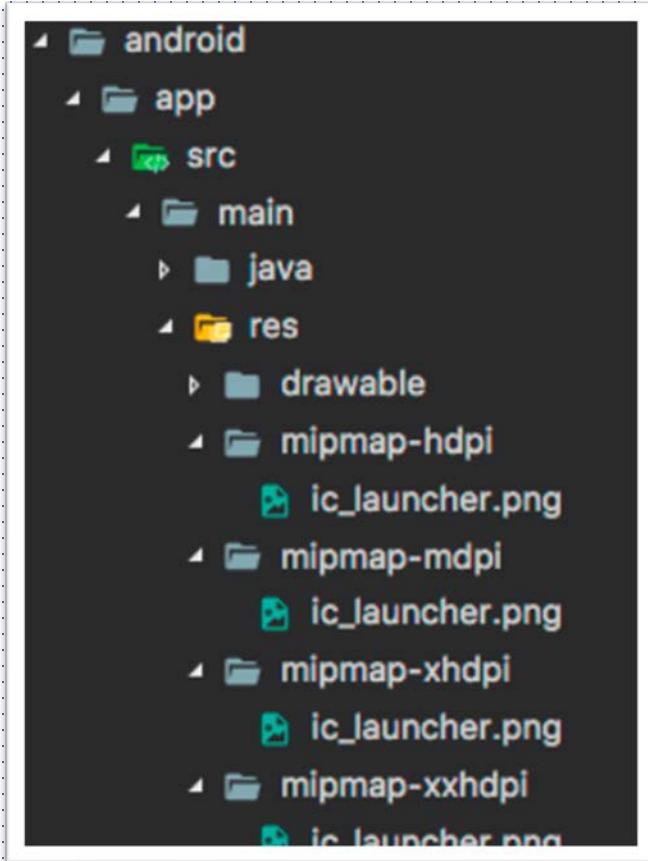
class MyWidget extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return FutureBuilder(
            future: DefaultAssetBundle.of(context).loadString('assets/config.json'),
            builder: (context, snapshot) => Text(snapshot.data ?? ''),
        );
    }
}
```

[DefaultAssetBundle](#) — это [InheritedWidget](#), который позволяет получить текущий [AssetBundle](#), а не использовать основной набор ресурсов по умолчанию [rootBundle](#). Этот подход позволяет родительскому виджету заменить [AssetBundle](#) во время выполнения, что бывает полезно для локализации или сценариев тестирования.

Платформенные ассеты

Иконка приложения

- Обновление иконки приложения Flutter выполняется таким же образом, как обновление иконки в нативных приложениях для Android.



В корневом каталоге проекта [Flutter](#) перейдите в [android/app/src/main/res](#). Различные папки ресурсов в формате [bitmap](#), такие как [mipmap-hdpi](#), уже содержат изображения с именем [ic_launcher.png](#). Замените их на желаемые ресурсы, соблюдая рекомендуемый размер значков для каждой плотности экрана, указанный в [Android Developer Guide](#).



[Android Developer Guide](#)

Launch Screen

- Launch Screen — первый экран, который видит пользователь. Он появляется сразу после запуска приложения и сохраняется до тех пор, пока Flutter не отрисует первый кадр приложения. Обновляется он таким же образом, как в нативных приложениях.



Для добавления экрана запуска в приложении нужно перейти в папку [android/app/src/main](#). В файле [res/drawable/launch_background.xml](#) можно использовать [layer list drawable](#) XML-файл для кастомизации экрана запуска.

[Дополнительные сведения](#)