

在IDE中开发Flutter应用

Flutter插件在Android Studio或IntelliJ IDE中提供完全集成的开发体验。

- [安装和设置](#)
 - [Updating the plugins](#)
- [创建项目](#)
 - [创建一个新项目](#)
 - [从现有的源代码创建一个新的项目](#)
- [编辑代码和查看代码问题](#)
- [运行和调试](#)
 - [选择一个target](#)
 - [无断点运行](#)
 - [有断点运行](#)
- [快速编辑和加快开发周期](#)
- [高级调试](#)
 - [调试可视化布局问题](#)
 - [Debugging with Observatory](#)
- [Flutter代码提示](#)
 - [辅助 & 快速修正](#)
 - [用新的widget包装当前widget](#)
 - [用新的widget包装widget list](#)
 - [将 child 转为 children](#)
 - [实时模板](#)

- [IntelliJ键盘快捷键](#)
- [‘热重载’ vs ‘完全重启’](#)
- [在IntelliJ IDEA中编辑Android代码](#)
- [提示和技巧](#)
- [故障排除](#)
- [已知问题和反馈](#)

安装和设置

请按照[编辑器设置](#)说明安装Dart和Flutter插件。

Updating the plugins

对插件的更新将定期发布。当更新可用时，您在IntelliJ中会收到提示。

手动检查更新:

1. 打开 preferences (**IntelliJ IDEA>Check for Updates...** on macOS, **Help>Check for Updates...** on Linux).
2. 如果有 `dart` 后者`flutter`被列出, 更新它们.

创建项目

创建一个新项目

从Flutter入门应用程序模板创建一个新的Flutter IntelliJ项目：

1. 在IntelliJ中，在 ‘Welcome’ 窗口点击 **Create New Project** 或者在主界面 **File>New>Project...**
2. 在菜单中选择 **Flutter**，然后点击 **Next**.
3. 输入**Project name** 和 **Project location**
4. 点击 **Finish**.

从现有的源代码创建一个新的项目

To create a new Flutter IntelliJ project containing existing Flutter source code files:

创建一个包含现有Flutter源代码文件的新Flutter IntelliJ项目：

1. 在IntelliJ中，在 ‘Welcome’窗口点击**Create New Project** 或者在主界面 **File>New>Project...**

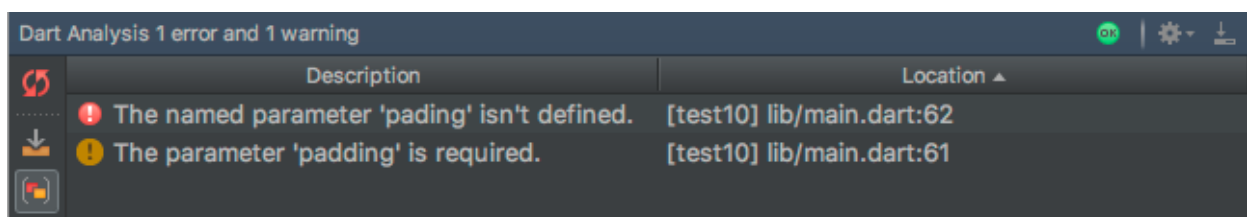
- **注意：不要使用New>Project from existing sources... 选项来创建。**

2. 在菜单中选择 **Flutter**，然后点击 **Next**.
3. 在 **Project location** 中输入，或者浏览选择现有的Flutter源代码文件目录
4. 点击 **Finish**.

编辑代码和查看代码问题

Dart插件执行代码分析，可以：

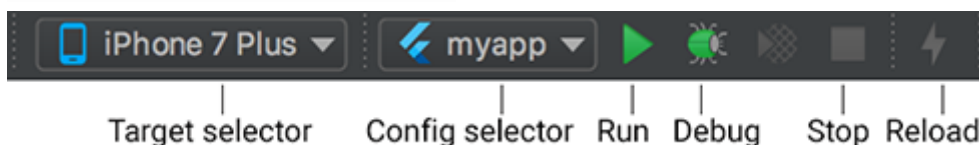
- 语法高亮显示.
- 基于丰富类型分析的代码补全.
- 导航到类型声明 (**Navigate>Declaration**), 查找类型使用的地方 (**Edit>Find>Find Usages**).
- 查看当前源代码的所有问题 (**View>Tool Windows>Dart Analysis**). 任何分析问题将在Analysis pane窗口中显示:



Dart Analysis pane

运行和调试

运行和调试由主工具栏控制：



Main IntelliJ toolbar

选择一个target

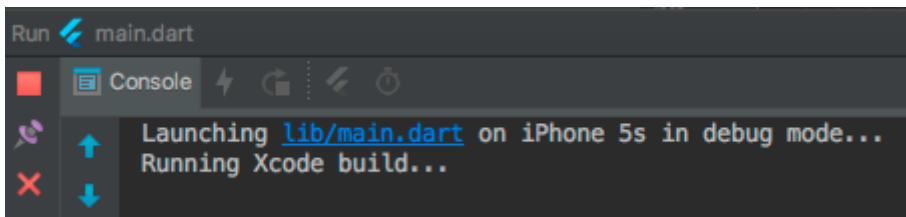
在IntelliJ中打开一个Flutter项目时，您应该在工具栏的右侧看到一组Flutter特定的按钮。

注意: 如果Run & Debug按钮被禁用，并且没有列出任何target，则Flutter没有发现任何连接的iOS或Android设备或模拟器。您需要连接设备或启动模拟器才能继续。

点击 **Flutter Target Selector** 下拉按钮，这将显示可用的设备列表，选择你想让你的应用运行的设备. 当您连接新的设备或启动新的模拟器时，里面会添加新的选项

无断点运行

点击 **运行** 图标，或者调用 **Run>Run**。底部的 **Run** 窗格中将会显示日志输出



Log pane

有断点运行

1. 如果需要，可在源代码中设置断点.
2. 点击工具栏的 **调试** 图标，或者调用 **Run>Debug**.
 - 底部的 **Debugger** 窗口将显示调用栈和变量.
 - 底部的 **Console** 窗口将显示详细的日志输出.
 - 调试基于默认的启动配置。要自定义这个，点击设备选择器右侧的下拉按钮，然后选择 **Edit configuration**

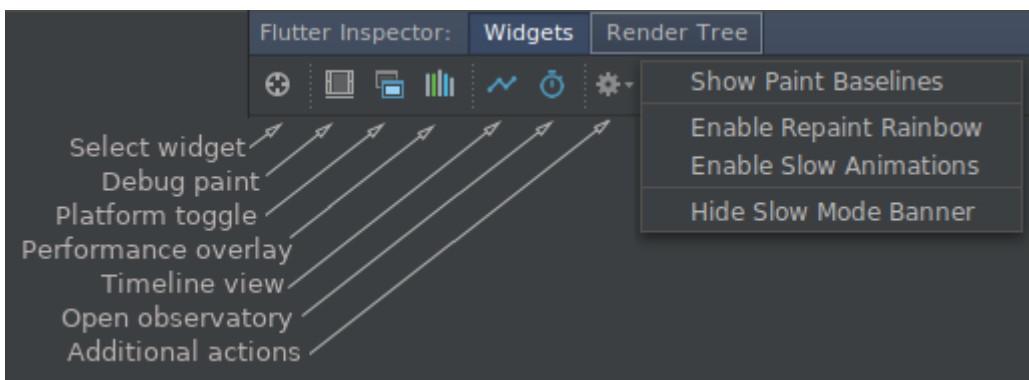
快速编辑和加快开发周期

Flutter提供了快速的开发周期循环，使您能够通过“热重新”功能在源码发生改变后几乎立即看到变更的效果。有关详细信息，请参阅[热重载Flutter应用程序](#)。

高级调试

调试可视化布局问题

要调试UI问题，请使用“Debug”启动应用程序，然后使用‘View > Tool Windows > Flutter Inspector’打开Flutter检查器工具窗口。



IntelliJ Flutter Inspector Window

这提供了许多调试工具; 有关这些详细信息，请参阅[调试Flutter Apps](#)

- ‘切换 Select Widget 模式’: 在设备上选择一个widget以在[Flutter Inspector](#)中对其进行检查。

- ‘切换 Debug Paint’: 显示Widget布局边界（包括边框、padding、对齐等）
- ‘切换 Platform’: 在Android或iOS渲染之间切换.
- ‘切换 Performance Overlay’: 显示GPU和CPU线程的性能图.
- ‘打开 Timeline 窗口’: 分析应用程序运行时的活动.
- ‘打开 Observatory’: Dart应用程序的分析器.

菜单中还有一些其他操作：

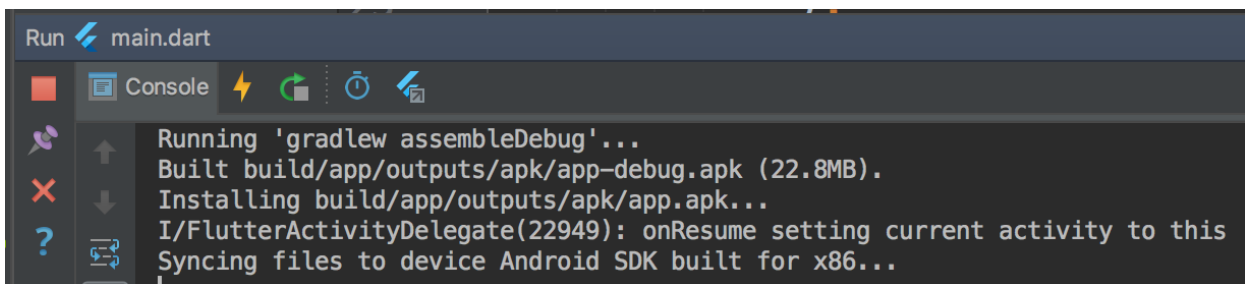
- ‘Show Paint Baselines’: 使每个RenderBox显示其基线
- ‘Enable Repaint Rainbow’: 重绘时在层上显示旋转颜色.
- ‘Enable Slow Animations’: 减慢动画以方便观察.
- ‘Hide Slow Mode Banner’: 使在运行调试版本时隐藏’slow mode’横幅

Debugging with Observatory

Observatory 是一个附带的基于HTML的用户界面的调试和分析工具. 详情请参考 [Observatory page](#).

打开 Observatory:

1. 以调试模式运行您的应用程序.
2. 在Debug面板中选择‘open observatory’ (见下面截图), 点击秒表⏱图标 (‘Open Observatory’).



Debugging panel

Flutter代码提示

辅助 & 快速修正

辅助是与特定代码标识符相关的代码更改。当光标放置在Flutter Widget标识符上时，可以使用其中的一些标识符，如黄色灯泡图标所示。可以通过单击灯泡或使用键盘快捷Alt-Enter来调用该辅助功能，如下所示：

```

    title: new Text(config.title),
  ),
  body: new Center(
    child: new Text(
      'Button tapped $_counter times${ _counter == 1 ? '' : 's' }.',
    ),
  ),
  floatingActionButton: new FloatingActionButton(
    onPressed: _incrementCounter,
    tooltip: 'Increment',
  ),
);

```

IntelliJ editing assists

快速修正是类似的，只有显示一段有错误的代码时，他们可以帮助您纠正它。它用一个红色灯泡表示。

辅助菜单中的几个功能：

用新的widget包装当前widget

这可以在您想要包装光标周围的widget时使用，例如，如果要将widget包装在一个Row或Column中。

用新的widget包装widget list

类似于上一条，但是用于包装现有widget 列表而不是单个widget。

将 child 转为 children

将child参数更改为children，并将参数值包装在列表中

实时模板

实时模板可用于加速输入常用的代码结构块。通过输入他们的'前缀'来调用它们，然后在代码补全窗口中选择它们：



```

void main() {
  runApp(null);
}

```

IntelliJ live templates

Flutter插件包含以下模板:

- 前缀`stless`: 创建一个`StatelessWidget`的子类.
- 前缀`stful`: 创建一个`StatefulWidget`子类并且关联到一个`State`子类.
- 前缀`stanim`: 创建一个`StatefulWidget`子类, 并且它关联的`State`子类包括一个 `AnimationController`

您还可以在 **Settings > Editor > Live Templates** 中自定义模板。

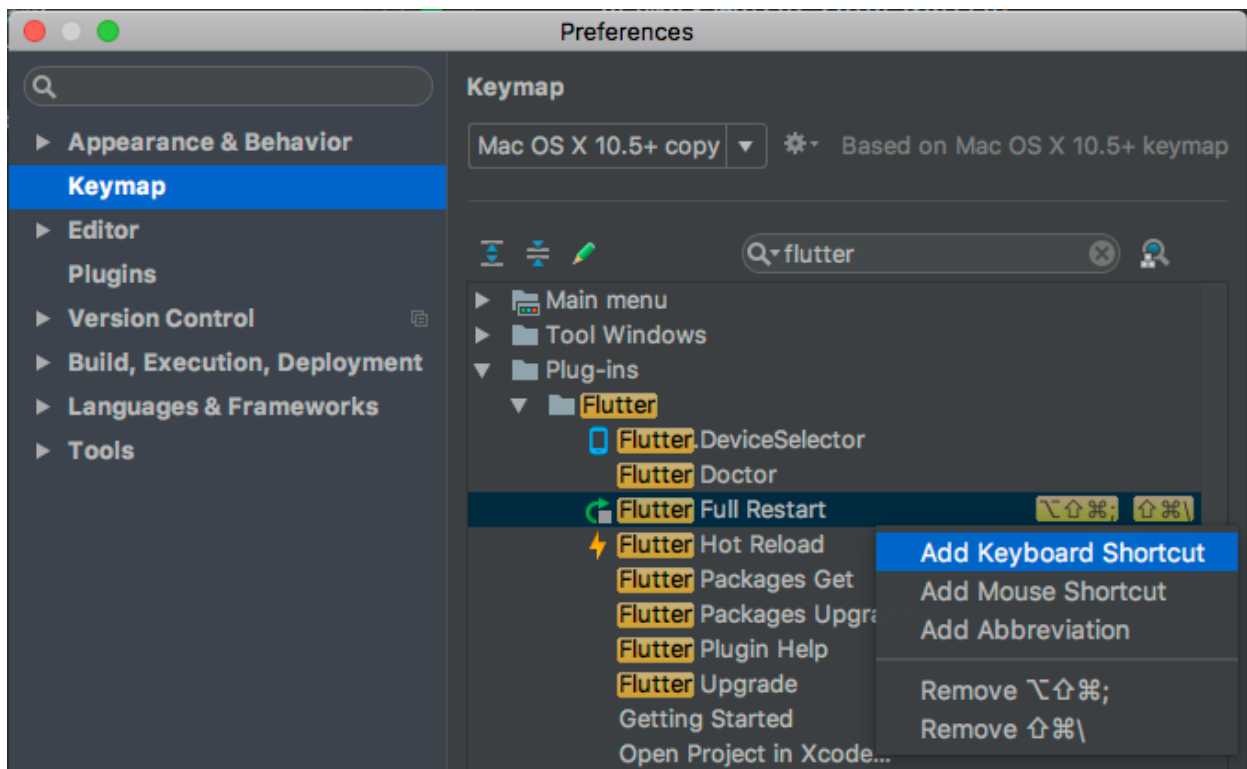
IntelliJ 键盘快捷键

热重载 (Hot Reload)

在Linux上 (IntelliJ 键盘映射默认为XWin) 和Windows键盘快捷键是`ctrl-alt-;`和`ctrl-\`。

在macOS上 (IntelliJ 键盘映射Mac OS X 10.5+ copy) , 键盘快捷键是`⌘-⌥-;`和`⌘-\`。

键盘映射可以在IntelliJ Preferences/Settings进行更改：选择`Keymap`，然后在右上角的搜索框中输入“flutter”。右键单击要更改的绑定并添加键盘快捷键



IntelliJ Settings Keymap

‘热重载’ vs ‘完全重启’

热重载通过将更新的源代码文件注入正在运行的Dart VM (虚拟机) 中工作。这不仅包括添加新类，还包括向现有类添加方法和字段以及更改现有函数。尽管有几种类型的代码更改无法热重载：

- 全局变量初始化器.
- 静态字段初始化器.
- app的`main()`方法.

对于这些更改，您可以完全重新启动应用程序，而无需结束调试会话：

不要点击停止按钮; 只需重新单击运行按钮（如果在运行会话中）或调试按钮（如果在调试会话中），或者按住Shift键并单击“热重载”按钮

在IntelliJ IDEA中编辑Android代码

要在IntelliJ IDEA中编辑Android代码，您需要配置Android SDK的位置：

1. 在Preferences->Plugins中, 启用 **Android Support**（如果你还没有启用）.
2. 右键单击项目视图中的android文件夹，然后选择**Open Module Settings**。
3. 在 **Sources** 选项卡中, 找到 **Language level** 字段, 然后选择 ‘8’或更高级别
4. In **Dependencies** tab, locate the **Module SDK** field, and select an
5. 在**Dependencies**选项卡中，找到**Module SDK**字段，然后选择一个Android SDK。如果没有列出SDK，请单击**New...**并指定Android SDK的位置。请确保选择与Flutter使用的Android SDK相匹配的Android SDK（如flutter doctor所提示的）。
6. 点击**OK**.

提示和技巧

请查看这些‘cheat sheets’:

- [Flutter IntelliJ cheat sheet, MacOS version](#)
- [Flutter IntelliJ cheat sheet, Windows & Linux version](#)

故障排除

已知问题和反馈

[Flutter plugin README](#)文件中记录了可能影响您的体验的重要已知问题。

所有已知的错误都会持续跟踪：

- Flutter 插件: [GitHub issue 跟踪](#).
- Dart 插件: [JetBrains YouTrack](#).

我们非常欢迎有关错误/问题和功能请求的反馈。在提交新问题之前，请：

- 在问题跟踪中快速搜索以查看该问题是否已被跟踪
- 确保你已经 [更新](#) 到了最新版本的插件

提交新issue时，请包括[flutter doctor](#)的输出