文档支持语言

pt_BR **en_US** **zh_CN - 本文件**

GetX™ 框架的官方 CLI。

```
// 安装:
pub global activate get_cli
// 使用本命令需要设置系统环境变量: [FlutterSDK安装目录]\bin\cache\dart-sdk\bin 和 [FlutterSDK安装目录]\.pub-cache\bin

flutter pub global activate get_cli

// 在当前目录创建一个 Flutter 项目:
// 注: 默认使用文件夹名称作为项目名称
// 你可以使用 `get create project:my_project` 给项目命名
// 如果项目名称有空格则使用 `get create project:"my cool project"`
get create project

// 在现有项目中生成所选结构:
get init

// 创建页面:
// (页面包括 controller, view, 和 binding)
// 注: 你可以随便命名, 例如: `get create page:login`
get create page:home

// 在指定文件夹创建新 controller:
// 注: 你无需引用文件夹, Getx 会自动搜索 home 目录,
// 并把你的controller放在那儿
get create controller:dialogcontroller on home

// 在指定文件夹创建新 view:
// 注: 你无需引用文件夹,Getx 会自动搜索 home 目录,
// 并把你的 view 放在那儿
get create view:dialogview on home

// 在指定文件夹创建新 provider:
get create provider:user on home

// 生成国际化文件:
// 注: 你在 'assets/locales' 目录下的翻译文件应该是json格式的
get generate locales assets/locales

// 生成 model 类:
// 注: 'assets/models/' 目录下的模板文件应该是json格式的
// 注: on  == 输出文件夹
// Getx 会自动搜索 home 目录,
// 并把你的 model 放在那儿
get generate model on home with assets/models/user.json

//生成无 provider 的 model
get generate model on home with assets/models/user.json --skipProvider

//注: URL 必须返回json
get generate model on home from "https://api.github.com/users/CpdnCristiano"

// 为你的项目安装依赖:
get install camera

// 为你的项目安装多个依赖:
get install http path camera

// 为你的项目安装依赖(指定版本号):
get install path:1.6.4

// 你可以为多个依赖指定版本号

// 为你的项目安装一个dev依赖(dependencies_dev):
get install flutter_launcher_icons --dev

// 为你的项目移除一个依赖:
get remove http

// 为你的项目移除多个依赖:
get remove http path
```

```
// 更新 CLI:
get update
// 或 `get upgrade`

// 显示当前 CLI 版本:
get -v
// 或 `get -version`

// 帮助
get help
```

探索 CLI

让我们看看 CLI 都有啥命令吧

新建项目

```
get create project
```

用来新建一个项目, 你可以在 Flutter 和 get_server里选一个, 创建默认目录之后, 它会运行一个 `get init`

初始化

```
get init
```

这条命令要慎用，它会覆盖 lib 文件夹下所有内容。它允许你在两种结构中二选一, getx_pattern 和 clean.

创建 Page

```
get create page:name
```

这条命令允许你创建模块，建议选择getx_pattern的人使用。

创建 view, controller 和 binding 文件, 除了自动添加路由。

你可以在一个模块内创建另一个模块。

```
get create page:name on other_module
```

当你创建一个项目，并且用 `on` 创建一个页面，CLI 会使用children pages.

创建 Screen

```
get create screen:name
```

和 `create page` 类似, 但更适合使用 Clean 的人。

创建 controller

```
get create controller:dialog on your_folder
```

在指定目录创建 controller

带选项使用 你可以创建一个模板文件, 用你喜欢的方式

运行

```
get create controller:auth with examples/authcontroller.dart on your_folder
```

或者使用 URL *运行*

```
get create controller:auth with 'https://raw.githubusercontent.com/jonataslaw/get_cli/master/samples_file/controller.dart.e
```

输入:

```
@import

class @controller extends GetxController {
  final  email = ''.obs;
  final  password = ''.obs;
  void login() {
  }
```

```
}
```

输出:

```
import 'package:get/get.dart';

class AuthController extends GetxController {
  final email = ''.obs;
  final password = ''.obs;
  void login() {}
}
```

创建 view

```
get create view:dialog on your_folder
```

在指定目录创建 view

生成国际化文件

在 assets/locales 目录创建 json 格式的语言文件

输入:

zh_CN.json

```
{
  "buttons": {
    "login": "登录",
    "sign_in": "注册",
    "logout": "注销",
    "sign_in_fb": "用 Facebook 登录",
    "sign_in_google": "用 Google 登录",
    "sign_in_apple": "用 Apple 登录"
  }
}
```

en_US.json

```
{
  "buttons": {
    "login": "Login",
    "sign_in": "Sign-in",
    "logout": "Logout",
    "sign_in_fb": "Sign-in with Facebook",
    "sign_in_google": "Sign-in with Google",
    "sign_in_apple": "Sign-in with Apple"
  }
}
```

运行:

```
get generate locales assets/locales
```

输出:

```
abstract class AppTranslation {

  static Map<String, Map<String, String>> translations = {
    'en_US' : Locales.en_US,
    'zh_CN' : Locales.zh_CN,
  };

}
abstract class LocaleKeys {
  static const buttons_login = 'buttons_login';
  static const buttons_sign_in = 'buttons_sign_in';
  static const buttons_logout = 'buttons_logout';
  static const buttons_sign_in_fb = 'buttons_sign_in_fb';
  static const buttons_sign_in_google = 'buttons_sign_in_google';
  static const buttons_sign_in_apple = 'buttons_sign_in_apple';
}

abstract class Locales {

  static const en_US = {
   'buttons_login': 'Login',
   'buttons_sign_in': 'Sign-in',
   'buttons_logout': 'Logout',
   'buttons_sign_in_fb': 'Sign-in with Facebook',
   'buttons_sign_in_google': 'Sign-in with Google',
```

```
    'buttons_sign_in_apple': 'Sign-in with Apple',
  };
  static const zh_CN = {
    'buttons_login': 'Entrar',
    'buttons_sign_in': 'Cadastrar-se',
    'buttons_logout': 'Sair',
    'buttons_sign_in_fb': '用 Facebook 登录',
    'buttons_sign_in_google': '用 Google 登录',
    'buttons_sign_in_apple': '用 Apple 登录',
  };

}
```

现在只需要在 GetMaterialApp 中加入

```
    GetMaterialApp(
      ...
      translationsKeys: AppTranslation.translations,
      ...
    )
```

例：生成 model

创建json model 文件assets/models/user.json

输入:

```
{
  "name": "",
  "age": 0,
  "friends": ["", ""]
}
```

运行:

```
get generate model on home with assets/models/user.json
```

输出:

```
class User {
  String name;
  int age;
  List<String> friends;

  User({this.name, this.age, this.friends});

  User.fromJson(Map<String, dynamic> json) {
    name = json['name'];
    age = json['age'];
    friends = json['friends'].cast<String>();
  }

  Map<String, dynamic> toJson() {
    final Map<String, dynamic> data = new Map<String, dynamic>();
    data['name'] = this.name;
    data['age'] = this.age;
    data['friends'] = this.friends;
    return data;
  }
}
```

拆分不同类型文件

有一天有个用户问我，是否可能修改一下最终文件名，他发现 `my_controller_name.controller.dart` 比 CLI 生成的默认文件 `my_controller_name_controller. dart` 更具有可读性，考虑到像他这样的用户，我加了个选项，可以让你选择你自己的分隔符，只需要在你的 pubsepc.yaml 里这样写

例子:

```
get_cli:
  separator: "."
```

你的 import 乱不乱?

为了帮你管理你的 import 我加了个新命令: `get sort` , 除了帮你排序整理 import, 这条命令还帮你格式化 dart 文件。感谢 dart_style. `get sort` 会用 separator 重命名所有文件。如果不想重命名文件，使用 `--skipRename` 。

如果你喜欢用相对路径写 import, 使用 `--relative` 选项. get_cli 会自动转换。

cli 国际化

CLI 现在有一套国际化系统。

如果你想把 CLI 翻译成你的语言:

1. 在 tranlations 目录创建一个你语言对应的json文件
2. 从 file 复制所有key, 然后翻译成你的语言
3. 发送你的 PR.

TODO:

- 自定义 model 支持
- 单元测试
- 优化生成结构
- 增加备份系统