搜索

添加新的自定义向导

如果您的团队正在处理一个或多个应用程序，则可能需要标准化团队成员创建项目和文件的方式。

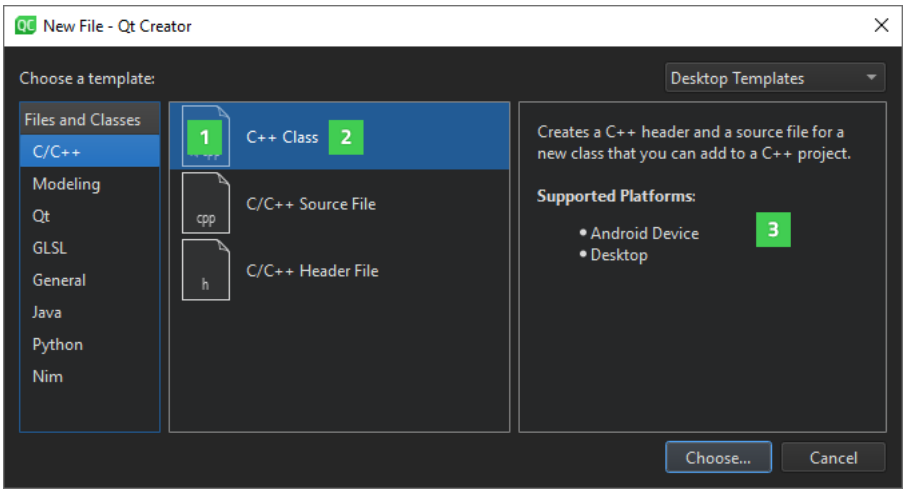
您可以创建 JSON 格式的自定义向导。它们存储在向导模板目录中，其中包含调用的 JSON 配置文件和所需的任何模板文件。配置文件包含指定有关向导的信息、可以使用的

要创建自定义向导，请将模板目录复制到共享目录或本地用户的设置目录（以新名称）。然后更改文件中的向导 ID。`wizard.json`

您可以在设置目录中为模板创建子目录。标准向导按类型组织到子目录中，但您可以将向导目录添加到您喜欢的任何目录中。文件夹层次结构不会影响向导的显示顺序。

要与其他用户共享向导，您可以创建向导目录的存档，并指示收件人将其解压缩到Qt Creator从中搜索向导的目录中。

Qt Creator 显示它在"新建**项目**"和"**新建文件**"对话框中找到的向导。对于每个向导，将显示一个图标 （1）、一个显示名称 （2） 和一个说明 （3）。



向导类型

在项目向导中，可以指定项目中所需的文件。可以添加向导页，以允许开发人员指定项目的设置。

文件向导与此类似，但不包含任何项目文件。

定位向导

Qt Creator在以下位置搜索向导：

- 共享目录：
  - 在视窗上：`share\qtcreator\templates\wizards`
  - 在 Linux 上：`share/qtcreator/templates/wizards`
  - 在 macOS 上：`Qt Creator.app/Contents/Resources/templates/wizards`
- 本地用户的设置目录：
  - 在视窗上：`%APPDATA%\QtProject\qtcreator\templates\wizards`
  - 在 Linux 和 macOS 上：`$HOME/.config/QtProject/qtcreator/templates/wizards`

向导开发提示

为某些帮助程序操作分配键盘快捷键并打开详细输出。

将操作映射到键盘快捷键

Qt Creator有一些可以改进向导开发过程的操作。默认情况下，这些不绑定到任何键盘快捷键，因此无法触发。要启用它们，请在"**编辑**>**首选项**>**环境**>**键盘向导**>中分配键盘

以下操作有助于向导开发：

恢复出厂设置.重置	触发此操作会使Qt Creator忘记所有向导工厂，导致它在打开**File>New Project**时重新加载所有向导定义。这样，您可以避免重新启动Qt Creat

详细输出

对于向导开发，我们建议您使用参数启动Qt Creator，以接收Qt Creator能够找到并解析文件的确认。详细模式显示有关语法错误的信息，语法错误是编辑向导时可能遇到的最

在详细模式下，每个正确设置的向导都会生成以下行的输出：

```
Checking "/home/jsmith/.config/QtProject/qtcreator/templates/wizards/mywizard"
for wizard.json.
* Configuration found and parsed.
```

输出包括检查文件目录的名称。如果未找到该文件，则不显示该消息。`wizard.json`

如果文件包含错误（如无效的图标路径），则会显示以下类型的消息：

```
Checking "/home/jsmith/.config/QtProject/qtcreator/templates/wizards/mywizard"
for wizard.json.
* Configuration found and parsed.
* Failed to create: Icon file
"/home/jsmith/.config/QtProject/qtcreator/templates/wizards/mywizard/../..
/global/genericfilewizard.png" not found.
```

有关命令行参数的详细信息，请参阅使用命令行选项。

将向导集成到构建中

要将向导集成到Qt Creator中并将其作为Qt Creator构建的一部分交付，请将向导文件放在Qt Creator源中。然后选择"**生成**>**运行 CMake**或"**运行 qmake**"，具体取决于所使用的生分。

如果不运行CMake或qmake，则新向导将不会显示，因为它不存在于运行新构建的Qt Creator的构建目录中。它从未被复制到那里，因为CMake或qmake没有通知构建工具，如

基本上，CMake 和 qmake 会生成一个固定的文件列表，以从源目录复制到在运行时检查向导的构建目录的子目录。因此，您需要运行 CMake 或 qmake 或执行**Factory.Reset**函

在向导中使用变量

您可以在 JSON 配置文件和模板源文件中的字符串中使用变量 （）。一组变量由向导及其页面预定义。您可以通过在文件中的部分中定义变量键名称和值来引入新变量作为快

有一个特殊的变量，它计算给定的 JavaScript 表达式并将生成的 JavaScript 值转换为字符串。在 JavaScript 表达式中，您可以引用向导定义的变量。返回的 JavaScript 对象具有

在需要布尔值并给出字符串的地方，空字符串和字符串被视为其他任何内容。`"false"falsetrue`

本地化向导

如果设置名称以前缀开头，则该值对用户可见，应进行翻译。如果新向导包含在Qt Creator源中，则可翻译的字符串将显示在Qt Creator翻译文件中，并且可以作为Qt Creator的

```
"trDisplayName": { "C": "default", "en_US": "english", "de_DE": "deutsch" }
```

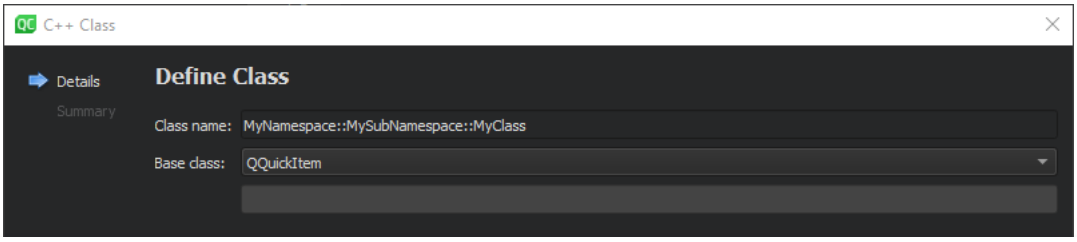
例如：

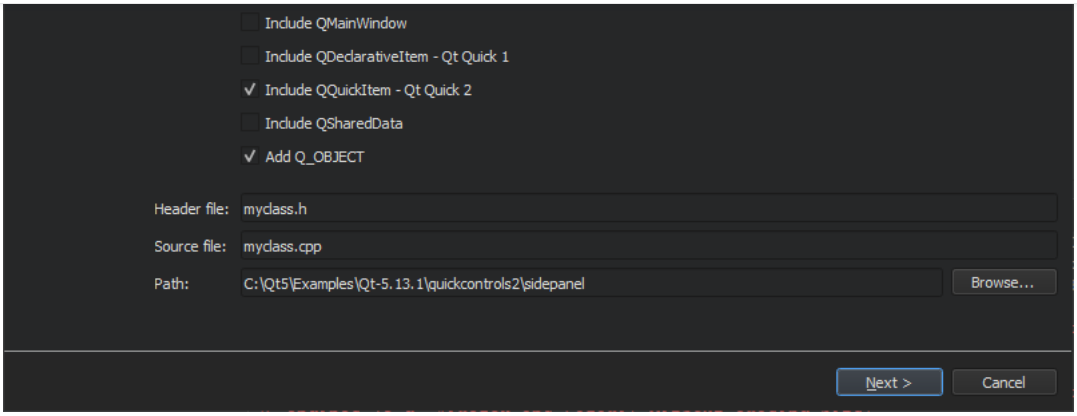
```
"trDisplayName": { "C": "Project Location", "en_US": "Project Location", "de_DE": "Projekt Verzeichnis" }
```

创建向导

Qt Creator包含用于添加类、文件和项目的向导。您可以使用它们作为添加自己的向导的基础。我们使用C++向导来解释该过程以及 .json 文件中的部分和设置。

在此示例中，我们在共享目录中创建向导目录，并将其集成到Qt Creator构建系统中，以便它可以与Qt Creator二进制文件一起部署作为构建的一部分。





有关可以添加的页面和微件及其支持的属性的详细信息，请参阅可用页面和可用微件。

创建基于 JSON 的C++类向导：

1. 使用参数启动Qt Creator，以在向导开发期间接收反馈。有关详细信息，请参阅详细输出。`-customwizard-verbose`
2. 为"**检查**"和"**恢复出厂设置**"操作的键盘快捷键，如向导开发提示中所述。
3. 制作副本并重命名。例如`share/qtcreator/templates/wizards/classes/cppshare/qtcreator/templates/wizards/classes/mycpp`
4. 使用**Factory.Reset**操作使向导显示在File>**New File**中，而无需重新启动Qt Creator。
5. 打开向导配置文件，进行编辑：`wizard.json`
   - 以下设置确定向导的类型及其在"**新建文件**"对话框中的位置：

```
"version": 1,
"supportedProjectTypes": [ ],
"id": "A.Class",
"category": "O.C++",
```

     - `version`是文件内容的版本。请勿修改此值。
     - `supportedProjectTypes`是一个可选设置，可用于在向现有项目添加新生成目标时筛选向导。例如，在向现有 qmake 项目添加新目标时，应仅提供生成

       可能的值是Qt Creator支持的构建系统，或者如果未指定构建系统：,,,,,（qmake项目），
       `UNKNOWN_PROJECTAutotoolsProjectManager.AutotoolsProjectCMakeProjectManager.CMakeProjectGenericProjectManager.Gene`
     - `id`是向导的唯一标识符。向导按 ID 中的字母顺序排序。可以使用前导字母指定向导的位置。必须始终更改此值。例如。`categoryB.MyClass`

       此信息在向导中可用。`%\{id\}`
     - `category`是将向导放入列表中的类别。可以使用前导字母指定类别在"**新建文件**"对话框中列表中的位置。

       此信息在向导中可用。`%\{category\}`
   - 以下设置指定"**新建文件**"对话框中显示的图标和文本：

```
"trDescription": "Creates a C++ header and a source file for a new class that you can add to a C++ project.",
"trDisplayName": "C++ Class",
"trDisplayCategory": "C++",
"iconText": "h/cpp",
"enabled": "%{JS: value('Plugins').indexOf('CppEditor') >= 0}",
```

     - `trDescription`何时显示在最右侧的面板中。`trDisplayCategory`

       此信息在向导中可用。`%\{trDescription\}`
     - `trDisplayName`何时显示在中间面板中。`trDisplayCategory`

       此信息在向导中可用。`%\{trDisplayName\}`
     - `trDisplayCategory`显示在"**新建**文件"对话框中的"**文件和类**"下。

       此信息在向导中可用。`%\{trDisplayCategory\}`
     - `icon`显示在中间面板旁边 何时被选中。建议指定相对于 wizard.json 文件的路径，但也可以使用绝对路径。省略此值以使用向导类型的默认图标。`trDispl`
     - `iconText`确定默认文件图标的文本叠加。
     - `iconKind`确定图标是否为主题。
     - `image`指定显示在下方的图像（例如屏幕截图）的路径。`trDescription`
     - `featuresRequired`指定向导所依赖的 Qt 创建器功能。如果缺少必需的功能，则向导将被隐藏。例如，如果没有套件设置了Qt版本，则隐藏了基于qmake

       使用如果您需要表达更复杂的逻辑来决定您的向导是否可用。`enabled`

       此信息在向导中可用。`%\{RequiredFeatures\}`
     - `featuresPreferred`指定要预选的生成和运行工具包。

- platformIndependent设置为如果所有目标平台都支持该向导。默认情况下，它设置为 。truefalse
- enabled进行评估以确定向导是否列在“**文件>新建项目**”或“**新建文件**”中，之后已检查。featuresRequired

  默认值为。true

- 该部分包含具有*键*和*值*属性的对象数组。除了预定义的变量之外，您还可以定义自己的变量，以便在配置和模板源文件中使用。例如，在C++类创建向导中使用以

```
"options":
[
    { "key": "TargetPath", "value": "%{Path}" },
    { "key": "HdrPath", "value": "%{Path}/%{HdrFileName}" },
    { "key": "SrcPath", "value": "%{Path}/%{SrcFileName}" },
    { "key": "CN", "value": "%{JS: Cpp.className(value('Class'))}" },
    { "key": "Base", "value": "%{JS: value('BaseCB') === '' ? value('BaseEdit') : value('BaseCB')}" },
    { "key": "isQObject", "value": "%{JS: (value('Base') === 'QObject' || value('Base') === 'QWidget' || value('Base') === 'QMainWindo
    { "key": "GUARD", "value": "%{JS: Cpp.classToHeaderGuard(value('Class'), Util.suffix(value('HdrFileName'))}" },
    { "key": "SharedDataInit", "value": "%{JS: value('IncludeQSharedData') ? 'data(new %{CN}Data)' : '' }" }
],
```

此部分是可选的。有关变量的更多示例，请参阅其他向导的文件。wizard.json

- 该部分指定向导页。使用的页面取决于向导类型。您可以将标准页面添加到向导或使用可用的小组件创建新页面。以下设置指定页面的显示名称、标题和类型：p

```
"pages":
[
    {
        "trDisplayName": "Define Class",
        "trShortTitle": "Details",
        "typeId": "Fields",
        "data" :
        [
            {
                "name": "Class",
                "trDisplayName": "Class name:",
                "mandatory": true,
                "type": "LineEdit",
                "data": {
                    "trPlaceholder": "Fully qualified name, including namespaces",
                    "validator": "(?:(?:[a-zA-Z_][a-zA-Z_0-9]*::)*[a-zA-Z_][a-zA-Z_0-9]*|)",
                    "completion": "namespaces"
                    }
    },
            ...
]
```

  - typeId指定要使用的页面：,,,,,或。FieldsFileFormKitsProjectVcsConfigurationVcsCommandSummary

    代码中使用的整页 ID 由前缀 .有关页面的详细信息，请参阅可用页面。typeId"PE.Wizard.Page."

  - trDisplayName指定页面的标题。默认情况下，使用页面标题。
  - trShortTitle指定向导边栏中使用的标题。默认情况下，使用页面标题。
  - trSubTitle指定页面的副标题。默认情况下，使用页面标题。
  - index是指定页面 ID 的整数值。如果不设置，它会自动分配。
  - enabled设置为显示页面并将其隐藏。truefalse
  - data指定向导页。在C++向导中，它指定页面和页面。该页面包含,,,和小部件。有关小组件的更多信息，请参阅可用小组件。FieldsSummaryFieldsChe

- 该部分指定要添加到项目中的文件：generators

```
"generators":
[
    {
        "typeId": "File",
        "data":
        [
            {
                "source": "file.h",
                "target": "%{HdrPath}",
                "openInEditor": true
                "options": [
                    { "key": "Cpp:License:FileName", "value": "%{HdrFileName}" },
                    { "key": "Cpp:License:ClassName", "value": "%{CN}" }
                ]
            },
            {
                "source": "file.cpp",
                "target": "%{SrcPath}",
```

```
                    { "key": "Cpp:License:ClassName", "value": "%{CN}" }
                ]
            }
    ]
```

- typeId指定发生器的类型。目前，仅支持。FileScanner
- data允许进一步配置生成器。

向导可用的值

除了从文件本身获取的属性（参见创建向导）之外，Qt Creator还提供了一些信息供所有基于JSON的向导使用：wizard.json

- WizardDir是文件的绝对路径。wizard.json
- Features列出了通过Qt Creator中配置的任何套件提供的所有功能。
- Plugins包含在Qt Creator的当前实例中运行的所有插件的列表。
- Platform包含在"**文件>新建项目**"或"**新建文件**"对话框中选择的平台。此值可能为空。

仅当通过"**项目**"视图中节点的上下文菜单触发向导时，以下信息才可用：

- InitialPath与所选节点的路径。
- ProjectExplorer.Profile.Ids包含为所选节点的项目配置的工具包列表。

可用页面

您可以通过在 wizard.json 文件的部分中指定预定义页面来将预定义页面添加到向导中。pages

字段页面

字段页面具有值并包含微件。有关小组件定义的更多信息，请参阅可用的小组件。typeIdField

```
"pages":
[
    {
        "trDisplayName": "Define Class",
        "trShortTitle": "Details",
        "typeId": "Fields",
        "data" :
        [
            {
                "name": "Class",
                "trDisplayName": "Class name:",
                "mandatory": true,
                "type": "LineEdit",
                "data": {
                    "trPlaceholder": "Fully qualified name, including namespaces",
                    "validator": "(?:(?:[a-zA-Z_][a-zA-Z_0-9]*::)*[a-zA-Z_][a-zA-Z_0-9]*|)",
                    "completion": "namespaces"
                    }
            },
            ...
],
```

文件页

文件页面具有值。您可以省略键或为其分配一个空对象。typeIdFiledata

```
{
    "trDisplayName": "Location",
    "trShortTitle": "Location",
    "typeId": "File"
},
```

页面从向导中计算以设置初始路径和文件名。页面设置为要创建的文件的完整路径。InitialFileNameInitialPathTargetPath

表单页面

表单页面具有值。您可以省略键或为其分配一个空对象。typeIdFormdata

```
{
    "trDisplayName": "Choose a Form Template",
```

页面设置为包含表单内容的字符串数组。FormContents

包

工具包页面具有值。“工具包”页面的“部分”包含具有以下设置的对象：typeIdKitsdata

- projectFilePath以及项目文件的路径。
- requiredFeatures包含字符串或对象列表，这些字符串或对象描述工具包必须提供的功能才能在页面上列出。

  找到字符串时，必须设置此功能。改用对象时，将检查以下设置：

  - feature，它必须是一个字符串（将全部展开）。%\{<VariableName\}
  - condition，必须评估 Toorand 可用于从列表中对要素进行折扣。truefalse
- preferredFeatures具有与所需功能相同格式的列表。任何与中列出的所有功能相匹配的套件都将在此页面上预先选择。preferredFeaturesrequiredFeatur

```
{
    "trDisplayName": "Kit Selection",
    "trShortTitle": "Kits",
    "typeId": "Kits",
    "enabled": "%{IsTopLevelProject}",
    "data": { "projectFilePath": "%{ProFileName}" }
},
```

页面计算以设置在“**文件>新建项目**”或“**新建文件**”中选择的平台。%\{Platform\}

项目

项目页面具有值。它不包含任何数据或具有属性的对象，该属性将显示在生成的页面上.defaults to，这是用从文件的字段中获取的信息填充的。typeIdProjecttrDescri

```
{
    "trDisplayName": "Project Location",
    "trShortTitle": "Location",
    "typeId": "Project",
    "data": { "trDescription": "A description of the wizard" }
},
```

页面计算以设置初始项目路径。页面设置为项目目录。InitialPathProjectDirectoryTargetPath

总结

“摘要”页具有值。它不包含任何数据或空对象。typeIdSummary

```
{
    "trDisplayName": "Project Management",
    "trShortTitle": "Summary",
    "typeId": "Summary"
}
```

如果这是顶级项目，则页面设置为空字符串，否则。它设置为正在使用的版本控制系统的 ID。IsSubprojectyesVersionControl

VcsCommand

“VcsCommand ”页运行一组版本控制操作并显示结果。

此页面的部分采用具有以下键的对象：data

- vcsId使用要使用的版本控制系统的 ID。
- trRunMessage在版本控制运行时显示消息。
- extraArguments使用字符串或字符串列表定义传递给版本控制签出命令的额外参数。
- repository使用从版本控制系统签出的 URL。
- baseDirectory与要在其中运行结帐操作的目录。
- checkoutName与将创建的子目录一起保存签出的数据。
- extraJobs带有定义初始签出后要运行的其他命令的对象列表。这可用于进一步自定义存储库，例如添加其他远程存储库或设置版本控制系统的配置变量。

  每个由具有以下设置的对象定义：extraJob

  - skipIfEmpty将导致空参数从要运行的命令中静默删除（如果设置为 ）。默认为。truetrue

- arguments带有要传递的参数。command
- timeOutFactor可用于为长时间运行的命令提供比默认超时更长的超时。
- enabled将对其进行评估以决定是否实际执行此作业。

Vcs配置

“VcsConfiguration ”页要求用户配置版本控制系统，并且仅在配置成功后启用“**下一步**”按钮。

此页面的部分采用带有键的对象。此设置定义将配置的版本控制系统。datavcsId

可用小部件

您可以在字段页面上添加以下微件：

- 复选框
- 组合框
- 标签
- 行编辑
- 路径选择器
- 间隔
- 文本编辑

> **注意：**向导中仅支持以下部分中记录的设置。

为每个小组件指定以下设置：

- name指定小组件名称。此名称用作变量名称以再次访问该值。
- trDisplayName指定 UI 中可见的标签文本（ifis 不是）。spantrue
- type指定小部件的类型：,,,,,, 和。CheckBoxComboBoxLabelLineEditPathChooserSpacerTextEdit
- trToolTip指定使用鼠标悬停字段时要显示的工具提示。
- isComplete为所有字段进行评估，以确定向导的“**下一步**”按钮是否可用。所有字段都必须具有其评估才能发生这种情况。此设置默认为。isCompletetruetrue
- trIncompleteMessage在字段的评估结果时显示。isCompletefalse
- persistenceKey使用户选择持久化。该值被视为设置键。如果用户更改了小组件的默认值，则会存储用户提供的值，并在下次运行向导时成为新的默认值。
- visible设置为如果小组件可见，否则设置为。默认情况下，它设置为 。truefalsetrue
- enabled设置为如果小组件已启用，否则设置为 。默认情况下，它设置为 。truefalsetrue
- mandatory设置为如果此微件必须具有值才能启用“**下一步**”按钮。默认情况下，它设置为 。truetrue
- span设置为隐藏标签并跨越表单。默认情况下，它设置为 。有关更多信息，请参见在向导中使用变量。false
- data指定特定构件类型的其他设置，如以下各节所述。

复选框

```
{
    "name": "IncludeQObject",
    "trDisplayName": "Include QObject",
    "type": "CheckBox",
    "data":
    {
        "checkedValue": "QObject",
        "uncheckedValue": "",
        "checked": "%{JS: value('BaseCB') === 'QObject' ? 'true' : 'false'}"
    }
},
```

- checkedValue指定启用复选框时要设置的值。默认情况下，设置为。true
- uncheckedValue指定禁用复选框时要设置的值。默认情况下，设置为。false
- checked设置为如果启用该复选框，否则。truefalse

列表

> **注意：**组合框和图标列表类型都是列表类型的变体，因此它们可以具有相同的属性。

```
{
    "name": "BaseCB",
    "trDisplayName": "Base class:",
```

```
        "items": [ { "trKey": "<Custom>", "value": "" },
                   "QObject", "QWidget", "QMainWindow", "QDeclarativeItem", "QQuickItem" ]
    }
},
```

或

```
{
    "name": "ChosenBuildSystem",
    "trDisplayName": "Choose your build system:",
    "type": "IconList",
    "data":
    {
        "items": [
            { "trKey": "CMake", "value": "cmake", "icon": "cmake_icon.png", "trToolTip": "Building with CMake." },
            { "trKey": "Qbs", "value": "qbs", "icon": "qbs_icon.png", "trToolTip": "Building with Qbs." },
            { "trKey": "QMake", "value": "qmake", "icon": "qmake_icon.png", "trToolTip": "Building with QMake." }
        ]
    }
},
```

- items指定要放入列表类型的项的列表。该列表可以包含 JSON 对象和纯字符串。对于 JSON 对象，定义和配对，其中 是用户可见的列表项，包含与该项关联的数据。此
- index指定启用列表类型时要选择的索引。默认情况下，它设置为 。0
- disabledIndex指定在禁用列表类型时要显示的索引。

标签

```
{
    "name": "LabelQQC_2_0",
    "type": "Label",
    "span": true,
    "visible": "%{JS: value('CS') === 'QQC_2_0'}",
    "data":
    {
        "wordWrap": true,
        "trText": "Creates a deployable Qt Quick 2 application using Qt Quick Controls.",
    }
},
```

- wordWrap设置为启用自动换行。默认情况下，它设置为 。truefalse
- trText包含要显示的标签文本。

行编辑

```
{
    "name": "Class",
    "trDisplayName": "Class name:",
    "mandatory": true,
    "type": "LineEdit",
    "data": {
        "trPlaceholder": "Fully qualified name, including namespaces",
        "validator": "(?:(?:[a-zA-Z_][a-zA-Z_0-9]*::)*[a-zA-Z_][a-zA-Z_0-9]*|)",
        "completion": "namespaces"
    }
},
{
    "name": "BaseEdit",
    "type": "LineEdit",
    "enabled": "%{JS: value('BaseCB') === '' ? 'true' : 'false'}",
    "mandatory": false,
    "data":
    {
        "trText": "%{BaseCB}",
        "trDisabledText": "%{BaseCB}",
        "completion": "classes"
    }
},
```

- trText指定要显示的默认文本。
- trDisabledText指定要在禁用字段中显示的文本。

- validator指定要验证行编辑所依据的QRegularExpression。
- fixup指定用于修复字符串的变量。例如，将行编辑中的第一个字符转换为大写。
- isPassword是一个布尔值，它指定行编辑包含将被屏蔽的密码。
- historyId是一个键，它指定历史记录完成程序的项目列表的名称。此值 和 是互斥的，因此不要同时设置它们。completion
- restoreLastHistoryItem是一个布尔值，它指定将最后一个历史记录项自动设置为行编辑中的默认文本。此键只能设置为 true ifis 也设置。historyId

路径选择器

```
{
    "name": "Path",
    "type": "PathChooser",
    "trDisplayName": "Path:",
    "mandatory": true,
    "data":
    {
        "kind": "existingDirectory",
        "basePath": "%{InitialPath}",
        "path": "%{InitialPath}"
    }
},
```

- path指定所选路径。
- basePath指定查找相对于的基路径。
- kind定义要查找的内容：,,,,,,或。existingDirectorydirectoryfilesaveFileexistingCommandcommandany

间隔

```
{
    "name": "Sp1",
    "type": "Spacer",
    "data":
    {
        "factor": 2
    }
},
```

该设置指定要乘以此间隔条的布局间距的因子（整数）。factor

文本编辑

```
{
    "name": "TextField",
    "type": "TextEdit",
    "data" :
    {
        "trText": "This is some text",
        "richText": true
    }
}
```

- trText指定要显示的文本。
- trDisabledText指定禁用文本编辑时要显示的文本。
- richText设置为富文本，否则。truefalse

可用的发电机

Qt Creator支持两种不同的JSON向导生成器。

文件生成器

生成器需要在其部分中列出对象。每个对象定义一个要处理并复制到（或任何其他位置）的文件。Filedata%\{TargetPath\}

每个文件对象都可以采用以下设置：

- source指定模板文件相对于包含该文件的目录的路径和文件名。wizard.json

  如果未设置，则假定具有给定 inis 名称的文件是通过某种其他方式生成的。例如，这对于指定在从版本控制系统签出数据后作为项目打开的正确文件非常有用。sourc
- target指定生成的文件的位置（绝对位置或相对位置），该文件通常由向导页之一设置。%{TargetPath}

- isBinary将文件视为二进制文件，并防止在文件中执行替换（如果设置为）。此设置默认为。truefalse
- condition如果条件返回，则生成文件。此设置默认为。有关更多信息，请参见在向导中使用变量。truetrue

生成器在其部分中使用以下设置获取一个对象：Scannerdata

- binaryPattern是一个正则表达式，将与找到的所有文件名匹配。任何匹配项都将标记为二进制文件，并且将跳过此文件的模板替换。此设置默认为空模式，因此不
- subdirectoryPatterns是正则表达式模式的列表。将扫描与这些模式之一匹配的任何目录以及顶级目录。此设置默认为空列表，不会扫描任何子目录。

‹ 向项目添加库