

🔍 搜索

Qt 6.4 > Qt设计师手册 > 创建自定义小部件扩展

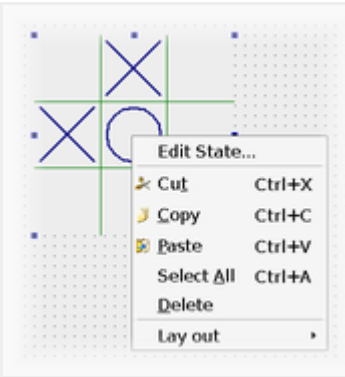
# 创建自定义小部件扩展

一旦你有了 *Qt Designer* 的自定义小部件插件，你就可以使用自定义小部件扩展在 *Qt Designer* 的工作空间中为其提供预期的行为和功能。

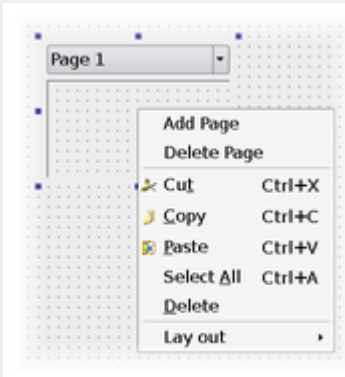
## 扩展类型

*Qt Designer* 中有几种可用的扩展类型。可以在同一模式中使用所有这些扩展，只需替换相应的扩展基类。

`QDesignerContainerExtension` 在实现自定义多页容器时是必需的。

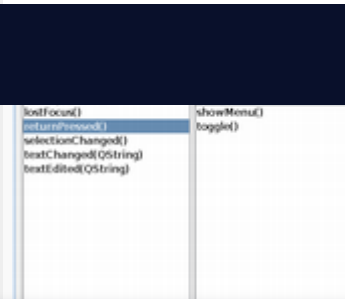


**QDesignerTaskMenuExtension**  
`QDesignerTaskMenuExtension` 对于自定义小部件很有用。它提供了一个扩展，允许您将自定义菜单项添加到 *Qt Designer* 的任务菜单中。  
**任务菜单扩展** 示例说明了如何使用此类。



**QDesignerContainerExtension**  
`QDesignerContainerExtension` 在实现自定义多页容器时是必需的。它提供了一个扩展，允许您在 *Qt Designer* 中添加和删除多页容器插件的页面。  
**容器扩展** 示例进一步说明了如何使用此类。

**注意：** 由于某些小部件（例如，`QTabWidget`）的实现方式，无法为它们添加自定义的每页属性。



**成员函数。**

Topics >

Property	Value
<b>QObject</b>	
objectName	tictactoe
<b>QWidget</b>	
modal	false
enabled	true
<b>geometry</b>	[60, 50, 200, 200]
sizePolicy	[Preferred, Preferred, 0, 0]
minimumSize	[0, 0]
maximumSize	[16777215, 16777215]
sizeIncrement	[0, 0]
baseSize	[0, 0]
palette	
font	[Sans Serif, 12]
cursor	[Arrow]

**QDesignerPropertySheetExtension, QDesignerDynamicPropertySheetExtension**  
 这些扩展类允许您控制小部件的属性在 *Qt Designer* 的属性编辑器中的显示方式。

*Qt Designer* 使用 **QDesignerPropertySheetExtension** 和 **QDesignerMemberSheetExtension** 类来提供其属性、信号和插槽编辑器。每当在其工作区中选择小部件时，*Qt Designer* 都会查询小部件的属性表扩展；同样，每当请求两个小部件之间的连接时，*Qt Designer* 都会查询小部件的成员表扩展。

**警告：** 所有小部件都具有默认属性和成员表。如果实现自定义属性表或成员表扩展，则自定义扩展将覆盖默认工作表。

## 创建扩展

若要创建扩展，必须继承 **QObject** 和相应的基类，并重新实现其函数。由于我们正在实现一个接口，因此我们必须确保使用扩展类定义中的 **Q\_INTERFACES ()** 宏将其告知元对象系统。例如：

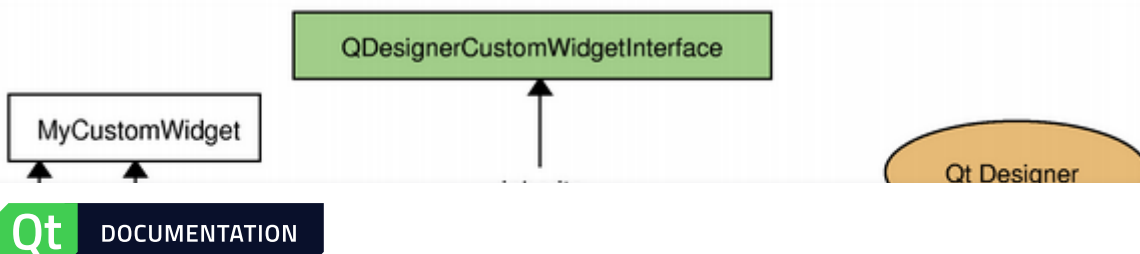
```
class MyExtension: public QObject,
                  public QDesignerContainerExtension
{
    Q_OBJECT
    Q_INTERFACES(QDesignerContainerExtension)
    ...
}
```

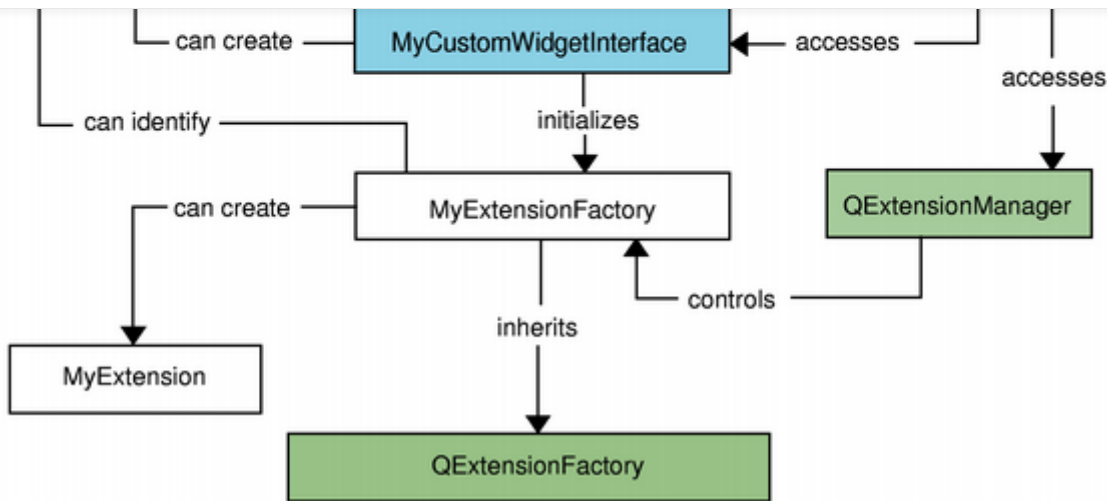
这使 *Qt 设计器* 能够使用 **qobject\_cast ()** 函数仅使用 **QObject** 指针查询支持的接口。

## 向Qt设计器公开扩展

在 *Qt Designer* 中，只有在需要扩展之前，才会创建扩展。因此，在实现扩展时，必须对 **QExtensionFactory** 进行子类，以创建一个能够创建扩展实例的类。此外，您必须向 *Qt Designer* 的扩展管理器注册您的工厂；扩展管理器处理扩展的构造。

当请求扩展时，*Qt Designer* 的扩展管理器将运行其注册的工厂，为每个工厂调用 **QExtensionFactory::createExtension ()**，直到找到能够为所选小部件创建请求扩展的工厂。然后，此工厂将创建扩展的实例。





## 创建扩展工厂

类提供了一个标准的扩展工厂，但它也可以用作自定义扩展工厂的接口。

目的是重新实现 `QExtensionFactory::createExtension()` 函数，使其能够创建扩展，例如 `MultiPageWidget` 容器扩展。

您可以创建一个新的 `QExtensionFactory` 并重新实现 `QExtensionFactory::createExtension()` 函数：

```

QObject *ANewExtensionFactory::createExtension(QObject *object,
    const QString &iid, QObject *parent) const
{
    if (iid != Q_TYPEID(QDesignerContainerExtension))
        return 0;

    if (MyCustomWidget *widget = qobject_cast<MyCustomWidget*>
        (object))
        return new MyContainerExtension(widget, parent);

    return 0;
}

```

或者您可以使用现有工厂，扩展 `QExtensionFactory::createExtension()` 函数以使工厂也能创建自定义扩展：

```

QObject *AGeneralExtensionFactory::createExtension(QObject *object,
    const QString &iid, QObject *parent) const
{
    MyCustomWidget *widget = qobject_cast<MyCustomWidget*>(object);

    if (widget && (iid == Q_TYPEID(QDesignerTaskMenuExtension))) {
        return new MyTaskMenuExtension(widget, parent);
    } else if (widget && (iid == Q_TYPEID(QDesignerContainerExtension))) {
        return new MyContainerExtension(widget, parent);
    }
}

```

}

## 访问Qt Designer的扩展管理器

在实现自定义控件插件时，必须对`QDesignerCustomWidgetInterface`进行子类化，以向*Qt Designer*公开您的插件。这在为Qt设计器创建自定义控件部分有更详细的介绍。扩展工厂的注册通常在`QDesignerCustomWidgetInterface::initialize()` 函数中进行：

```
void MyPlugin::initialize(QDesignerFormEditorInterface *formEditor)
{
    if (initialized)
        return;

    QExtensionManager *manager = formEditor->extensionManager();
    Q_ASSERT(manager != 0);

    manager->registerExtensions(new MyExtensionFactory(manager),
                               Q_TYPEID(QDesignerTaskMenuExtension));

    initialized = true;
}
```

`QDesignerCustomWidgetInterface::initialize()` 函数中的参数是指向*Qt Designer*当前`QDesignerFormEditorInterface`对象的指针。您必须使用`QDesignerFormEditorInterface::extensionManager()` 函数来检索*Qt Designer*扩展管理器的接口。然后使用`QExtensionManager::registerExtensions()` 函数来注册自定义扩展工厂。`formEditor`

## 相关示例

有关在*Qt Designer*中创建自定义控件扩展的更多信息，请参阅[任务菜单扩展](#)和[容器扩展](#)示例。

[< 为 Qt 设计器创建自定义小部件](#)

[Qt Designer的UI文件格式 >](#)

©2022 Qt有限公司 此处包含的文档贡献的版权归 他们各自的所有者。此处提供的文档根据自由软件基金会发布的GNU 自由文档许可证版本 1.3的条款进行许可。Qt和相应的徽标是Qt有限公司在芬兰和/或其他国家/地区的商标 全球。所有其他商标均为其各自所有者的财产。



联系我们



关于我们  
投资者  
编辑部  
职业  
办公地点

条款和条件  
开源  
常见问题

支持

支持服务  
专业服务  
合作 伙伴  
训练

对于客户

支持中心  
下载  
Qt登录  
联系我们  
客户成功案例

社区

为Qt做贡献  
论坛  
维基  
下载  
市场