Qt 项目 Doxygen 注释规范及使用建议

以下是为 Qt 项目定制的 Doxygen 注释规范，结合了您之前使用的风格和 Qt 的最佳实践。这套规范简洁明了，同时能生成高质量的文档：

### **一、文件头注释**

|  |
| --- |
| /\*\*  \* @file <文件名>.h  \* @brief <文件简要描述>  \* @details <详细描述，可省略>  \* @author <作者>  \* @date <创建/修改日期>  \* @version <版本号>  \*/ |

**示例：**

|  |
| --- |
| /\*\*  \* @file MyWidget.h  \* @brief 自定义Qt窗口部件  \* @author John Doe  \* @date 2023-05-15  \* @version 1.0  \*/ |

### **二、类注释**

|  |
| --- |
| /\*\*  \* @class <类名>  \* @brief <类的简要描述>  \* @details <详细描述，可省略>  \* @see <相关类或文档>  \*/  class <类名> : public <父类> {  Q\_OBJECT  // ...  }; |

**示例：**

|  |
| --- |
| /\*\*  \* @class MyWidget  \* @brief 自定义Qt窗口部件  \* @details 提供文本显示和编辑功能，继承自QWidget  \* @see QLabel, QLineEdit  \*/  class MyWidget : public QWidget {  Q\_OBJECT  // ...  }; |

### **三、Qt 属性注释**

|  |
| --- |
| /\*\*  \* @property <类名>::<属性名>  \* @brief <属性简要描述>  \* @details <详细描述，可省略>  \* @see <相关方法或属性>  \*/  Q\_PROPERTY(<类型> <属性名> READ <读方法> WRITE <写方法> NOTIFY <信号>) |

**示例：**

|  |
| --- |
| /\*\*  \* @property MyWidget::text  \* @brief 当前显示的文本  \* @details 支持HTML格式，文本变化时会发出textChanged信号  \* @see setText(), getText()  \*/  Q\_PROPERTY(QString text READ getText WRITE setText NOTIFY textChanged) |

### **四、方法 / 函数注释**

|  |
| --- |
| /\*\*  \* @brief <方法简要描述>  \* @param <参数名> <参数描述>  \* @return <返回值描述>  \* @see <相关方法>  \* @note <注意事项，可省略>  \* @sa <参考条目，可省略>  \*/ |

**示例：**

|  |
| --- |
| /\*\*  \* @brief 设置显示文本  \* @param text 要显示的文本内容，支持HTML格式  \* @return 是否成功设置  \* @see getText(), textChanged()  \* @note 过长的文本会自动换行  \*/  bool setText(const QString &text); |

### **五、信号与槽注释**

|  |
| --- |
| /\*\*  \* @brief <信号/槽简要描述>  \* @param <参数名> <参数描述>  \* @see <相关信号/槽>  \*/  signals:  void <信号名>(<参数列表>);  public slots:  void <槽名>(<参数列表>); |

**示例：**

|  |
| --- |
| signals:  /\*\*  \* @brief 文本改变时发出的信号  \* @param newText 新的文本内容  \* @see setText()  \*/  void textChanged(const QString &newText);  public slots:  /\*\*  \* @brief 清除显示文本  \* @see setText()  \*/  void clear(); |

### **六、枚举注释**

|  |
| --- |
| /\*\*  \* @enum <枚举名>  \* @brief <枚举简要描述>  \* @var <枚举名>::<枚举值>  \* <枚举值描述>  \*/  enum <枚举名> {  <枚举值1>, /\*\*< <枚举值1描述> \*/  <枚举值2>, /\*\*< <枚举值2描述> \*/  // ...  }; |

**示例：**

|  |
| --- |
| /\*\*  \* @enum MyWidget::Alignment  \* @brief 文本对齐方式  \* @var MyWidget::Alignment::Left  \* 左对齐  \* @var MyWidget::Alignment::Center  \* 居中对齐  \* @var MyWidget::Alignment::Right  \* 右对齐  \*/  enum Alignment {  Left, /\*\*< 左对齐 \*/  Center, /\*\*< 居中对齐 \*/  Right /\*\*< 右对齐 \*/  }; |

### **七、变量注释**

|  |
| --- |
| /\*\* <变量描述> \*/  <类型> <变量名>; |

**示例：**

|  |
| --- |
| /\*\* 当前字体大小，单位为像素 \*/  int m\_fontSize; |

### **八、Doxygen 配置建议**

在Doxyfile中添加以下配置，优化 Qt 项目文档生成：

|  |
| --- |
| # 启用Qt优化  OPTIMIZE\_OUTPUT\_QT = YES  QT\_AUTOBRIEF = YES  # 包含的文件类型  FILE\_PATTERNS = \*.h \*.cpp \*.ui  # 生成类图和关系图  HAVE\_DOT = YES  CLASS\_DIAGRAMS = YES  UML\_LOOK = YES  CALL\_GRAPH = YES  CALLER\_GRAPH = YES |

### **九、使用建议**

1. **工具集成**：在 Qt Creator 中添加自定义代码片段，快速插入规范注释。
2. **自动化检查**：使用静态分析工具（如 Clang-Tidy）检查注释覆盖率。
3. **定期生成文档**：通过批处理脚本（如之前提供的generate\_docs.bat）定期生成文档，确保文档与代码同步。

这套规范简洁且全面，能满足 Qt 项目的文档需求，同时生成的文档结构清晰，便于开发者查阅。