QEventLoop 事件循环

对于一个线程来说，其所拥有的事件队列是唯一的，但其所拥有的事件循环可以是多个，但绝对是嵌套关系，并且是只有当前QEventLoop被激活

让主线程等待100ms

// code snippet

QEventLoop eventLoop;

QTimer::singleShot(100,&eventLoop,SLOT(quit());

eventLoop.exec();

Q\_Q Q\_D

二进制兼容性

D 指针

保持一个库中的所有公有类的大小恒定的问题可以通过单独的私有指针给予解决。这个指针指向一个包含所有数据的私有数据结构体。这个结构体的大小可以随意改变而不会产生副作用，应用程序只使用相关的公有类，所使用的对象大小永远不会改变，它就是该指针的大小。这个指针就被称作D指针。

src/qtbase/src/corelib/kernel

qobject.h

qobject.cpp

qobject\_p.h

qpointer.h

qpointer.cpp

class QObject

{

Q\_DECLARE\_PRIVATE(QObject)

protected:

QScopedPointer<QObjectData> d\_ptr;

}

template <typename T>

static inline T\* qGetPtrHelper(T\* ptr){return ptr;}

// sample code

QScopedPointer<QObjectData> => QObjectPrivate\*

src/qtbase/src/corelib/global

qglobal.h

**Q\_DECLARE\_PRIVATE(Class)** \

inline Class##Private\* d\_func(){return reinterpret\_cast<Class##Private\*>( \

qGetPtrHelper(d\_ptr));}\

inline const Class##Private\* d\_func()const{ return reinterpret\_cast \

<Class##Private\*>(qGetPtrHelper(d\_ptr));} \

friend class Class##Private;

**Q\_DECLARE\_PUBLIC(Class)**

inline Class\* q\_func(){return static\_cast<Class\*>(q\_ptr);}

inline const Class\* q\_func()const{return static\_cast<Class\*>(q\_ptr);}

friend class Class;

**Q\_D(Class)** Class##Private\* const d = d\_func()

**Q\_Q(Class)** Class\* const q = q\_func()

信号槽

元对象系统 内省

QPaintEvent

QMenu

# UI Component

QWidget:public QObject,public QPaintDevice

{

protected:

virtual void paintEvent(QPaintEvent\* event);

}

void QWidget::paintEvent(QPaintEvent\* event)

{

}

CE :Control Element

class QFrame:public QWidget

{

protected:

void paintEvent(QPaintEvent\* event)override

{

QPainter paint(this);

drawFrame(&paint);

}

void drawFrame(QPainter\* p)

{

QStyleOptionFrame opt;

initStyleOption(&opt);

style()->drawControl(**QStyle::CE\_ShapedFrame**,&opt,p,this);

}

}

class QLabel:public QFrame

{

protected:

void paintEvent(QPaintEvent\* event)override

{

}

}

class QAbstractButton:public QWidget

{

protected:

void paintEvent(QPaintEvent\* event)override = 0;

}

class QCheckBox:public QAbstractButton

{

protected:

void paintEvent(QPaintEvent\* event)override

{

QStylePainter p(this);

QStyleOptionButton opt;

initStyleOption(&opt);

p.drawControl(**QStyle::CE\_CheckBox**,opt);

}

}

class QPushButton:public QAbstractButton

{

protected:

void paintEvent(QPaintEvent\* event)override

{

QStylePainter p(this);

QStyleOptionButton option;

initStyleOption(&option);

p.drawControl(**QStyle::CE\_PushButton**,option);

}

}

# Style

class QStyle:public QObject

{

}

class QCommonStyle:public QStyle

{

}

class QWindowsStyle:pulic QCommonStyle

{

}

class QWindowsXPStyle:public QWindowsStyle

{

}

class QWindowsVistaStyle:public QWindowsXPStyle

{

}

自定义外观

class BronzeStyle:public QWindowsStyle

{

}

# Box Model

qss