

# Plasma Widgets di KDE



plasma



K Desktop Environment



Code less.  
Create more.  
Deploy everywhere.



# Cos'è "Plasma" ?

**KDE 3**

Kicker

Kdesktop

SuperKaramba

**KDE 4**



# Cos'è “Plasma” ?

KDE lanciò Plasma nel 2005 con lo scopo di “rivitalizzare” l'interfaccia desktop (è “essenzialmente la stessa” dal 1984)

L'iniziativa cercava di rinnovare le fondamenta del desktop di KDE in previsione del rilascio di KDE 4 portando un cambiamento “radicale” nell'interfaccia tradizionalista del KDE 3.

Gli obiettivi chiave includevano l'unificazione di tre elementi fondamentali nel KDE: **Kicker** desktop panel, **Kdesktop** root window e **SuperKaramba** widget manager

in un'unica interfaccia “Plasma”.

Ne uscì fuori un framework che rendeva i widget molto semplici da scrivere e che assicurava una consistenza visuale e di accessibilità.

# Cos'è “Plasma” ?

- Panel, “Desktop”, Menu, Application Launcher
- Piattaforma di sviluppo Open Source “High-Level”
- Window Manager con supporto  
**Compositing** (effetti grafici desktop)
- Basato sulle Qt e su KDE
- Concepito con fondamenti di bellezza & usabilità

# In evidenza

- Accelerazione Hardware attraverso Qt
- I Widgets si possono scrivere in C++ o con linguaggi di scripting (JavaScript, HTML/CSS, Python, Ruby, ...)
- Target primario: mobile devices  
(small footprint, modularity, touchscreen, ...)  
\*le stesse Qt sono sviluppate da Nokia\*

# Plasmoids – Plasma's Widgets

- “Plasma” è la superficie su cui “poggiano” i widget
- Si posizionano su un “Panel” o sul desktop o su entrambi
- Si possono liberamente animare, ridimensionare, spostare, ruotare ...
- Condivisi online tramite apposita piattaforma
- “Ereditano” comportamenti e impostazioni di visualizzazione dal tema del KDE attuale

# Esempi

- “Desktop” widgets:
  - Application launcher menu, taskbar, desktop pager, ...
- System widgets:
  - Device notifier, power, network, notifications ...
- “Online” widgets:
  - RSS reader, widget meteo, dizionario, twitter, ...

# Esempi

- ~100 Plasma widgets 'nativi'
- Sono inoltre supportati:
  - Tutti i Google Gadgets
  - Tutto il contenuto Edge (E17)
  - Molti Widget della DashBoard di MacOS X

# In pratica ...

- Intercambiabilità assicurata:  
I Widgets scritti per un form factor (es. desktop) possono essere riutilizzati altrove (es. MID) poichè variano a runtime.
- I Widgets ed il loro contenuto può essere pacchettizzato e condiviso in rete.

# In pratica ...



# Pratica (quella seria)



Ma come si fanno i widget ??

# Come si crea un Widget (in C++)

I widget sono costituiti da almeno 4 file

- ✓ .desktop (informazioni)
- ✓ header
- ✓ implementazione
- ✓ CMakeLists.txt

# Come si crea un Widget (in C++)

.desktop File

Ogni Plasmoido necessita di un file .desktop per comunicare al sistema come dovrà essere gestito e quali sono le sue informazioni di base.

Il nostro file sarà: **plasma-applet-test.desktop**

**[Desktop Entry]**

**Name=Widget Widgettoso**

**Comment=Esempio di widget**

**Type=Service**

**ServiceTypes=Plasma/Applet**

**X-KDE-Library=plasma\_applet\_test**

**X-KDE-PluginInfo-Author=Pietro Lerro**

**X-KDE-PluginInfo-Email=lerro@startupsolutions.it**

**X-KDE-PluginInfo-Name=firstttest**

**X-KDE-PluginInfo-Version=0.1**

**X-KDE-PluginInfo-Website=http://www.startupsolutions.it/**

**X-KDE-PluginInfo-Category=Examples**

**X-KDE-PluginInfo-Depends=**

**X-KDE-PluginInfo-License=GPL**

**X-KDE-PluginInfo-EnabledByDefault=true**

# Come si crea un Widget (in C++)

.desktop File

Ogni Plasmoide necessita di un file .desktop per comunicare al suo contenitore come dovrà essere gestito e quali sono le sue informazioni di base.

Il nostro file sarà: **plasma-applet-test.desktop**

## [Desktop Entry]

**Name=Widget Widgettoso**

**Comment=Esempio di widget**

**Type=Service**

**ServiceTypes=Plasma/Applet**

**X-KDE-Library=plasma\_applet\_test**

**X-KDE-PluginInfo-Author=Pietro Lerro**

**X-KDE-PluginInfo-Email=lerro@startupsolutions.it**

**X-KDE-PluginInfo-Name=firstttest**

**X-KDE-PluginInfo-Version=0.1**

**X-KDE-PluginInfo-Website=http://www.startupsolutions.it/**

**X-KDE-PluginInfo-Category=Examples**

**X-KDE-PluginInfo-Depends=**

**X-KDE-PluginInfo-License=GPL**

**X-KDE-PluginInfo-EnabledByDefault=true**

Le informazioni chiave sono **X-KDE-Library** e **X-KDE-PluginInfo-Name**, esse rappresentano la “colla” che unisce la nostra classe con Plasma, senza queste, non partirebbe nulla.

# Come si crea un Widget (in C++)

.desktop File

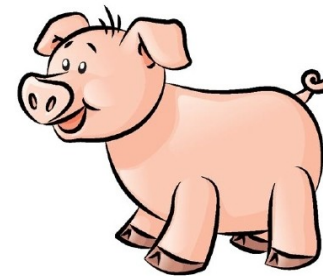
Ogni Plasmoido necessita di un file .desktop per comunicare al suo contenitore come dovrà essere gestito e quali sono le sue informazioni di base.

Il nostro file sarà: **plasma-applet-test.desktop**

## **X-KDE-PluginInfo-Category,**

è molto importante per la distribuzione dell'applet  
Avere una categoria di appartenenza.

Attualmente ci sono categorie predefinite, nel caso  
la nostra applet non ricadesse in una di quelle  
basta lasciar vuota questa property in modo che  
il sistema la assegni alla categoria "Miscellaneous"



**PIG**

**Plugin Info cateGory**

X-KDE-PluginInfo-Email=lerro@startupsolutions.it

X-KDE-PluginInfo-Name=firstttest

X-KDE-PluginInfo-Version=0.1

X-KDE-PluginInfo-Website=<http://www.startupsolutions.it/>

**X-KDE-PluginInfo-Category=Examples**

X-KDE-PluginInfo-Depends=

X-KDE-PluginInfo-License=GPL

X-KDE-PluginInfo-EnabledByDefault=true

<http://techbase.kde.org/Projects/Plasma/PIG>

# Come si crea un Widget (in C++)

## header file

```
// Here we avoid loading the header multiple times
#ifndef Tutorial1_HEADER
#define Tutorial1_HEADER
// We need the Plasma Applet headers
#include <KIcon>

#include <Plasma/Applet>
#include <Plasma/Svg>

class QSizeF;

// Define our plasma Applet
class PlasmaTutorial1 : public Plasma::Applet
{
    Q_OBJECT
public:
    // Basic Create/Destroy
    PlasmaTutorial1(QObject *parent, const QVariantList &args);
    ~PlasmaTutorial1();

    // The paintInterface procedure paints the applet to screen
    void paintInterface(QPainter *p,
        const QStyleOptionGraphicsItem *option,
        const QRect& contentsRect);
    void init();

private:
    Plasma::Svg m_svg;
    KIcon m_icon;
};

#endif
```

## - Struttura base -

- ✓ Costruttore / Distruttore
- ✓ Init
- ✓ PaintInterface

# Come si crea un Widget (in C++)

implementazione (1/2)

```
#include "plasma-tutorial1.h"
#include <QPainter>
#include <QFontMetrics>
#include <QSizeF>

#include <plasma/svg.h>
#include <plasma/theme.h>

PlasmaTutorial1::PlasmaTutorial1(QObject *parent, const QVariantList &args)
: Plasma::Applet(parent, args),
  m_svg(this),
  m_icon("document")
{
  m_svg.setImagePath("widgets/background");
  // this will get us the standard applet background, for free!
  setBackgroundHints(DefaultBackground);
  resize(200, 200);
}

PlasmaTutorial1::~PlasmaTutorial1()
{
  if (hasFailedToLaunch()) {
    // Do some cleanup here
  } else {
    // Save settings
  }
}
```

**Nel costruttore instancieremo tutte le risorse necessarie**

**Grazie al metodo `hasFailedToLaunch()` potremo sapere se l'applet è stata chiusa dall'utente o ha avuto problemi nel caricamento**

# Come si crea un Widget (in C++)

implementazione (2/2)

```
K_EXPORT_PLASMA_APPLET(<nome>, <classe>)
```

**Serve a collegare la nostra applet  
al file .desktop**

```
void PlasmaTutorial1::init()
{
    // A small demonstration of the setFailedToLaunch fun
    if (m_icon.isNull()) {
        setFailedToLaunch(true, "No world to say hello");
    }
}

void PlasmaTutorial1::paintInterface(QPainter *p,
    const QStyleOptionGraphicsItem *option, const QRect &contentsRect)
{
    p->setRenderHint(QPainter::SmoothPixmapTransform);
    p->setRenderHint(QPainter::Antialiasing);

    // Now we draw the applet, starting with our svg
    m_svg.resize((int)contentsRect.width(), (int)contentsRect.height());
    m_svg.paint(p, (int)contentsRect.left(), (int)contentsRect.top());

    // We place the icon and text
    p->drawPixmap(7, 0, m_icon.pixmap((int)contentsRect.width(),
(int)contentsRect.height()-14));
    p->save();
    p->setPen(Qt::white);
    p->drawText(contentsRect,
        Qt::AlignBottom | Qt::AlignHCenter,
        "Hello Plasmoid!");
    p->restore();
}

// This is the command that links your applet to the .desktop file
K_EXPORT_PLASMA_APPLET(tutorial1, PlasmaTutorial1)

#include "plasma-tutorial1.moc"
```

# Come si crea un Widget (in C++)

compilazione && installazione

## CMakeLists.txt

```
# Project Needs a name ofcourse
project(plasma-tutorial1)
```

```
# Find the required Libraries
find_package(KDE4 REQUIRED)
include(KDE4Defaults)
```

```
add_definitions(${QT_DEFINITIONS} ${KDE4_DEFINITIONS})
include_directories(
  ${CMAKE_SOURCE_DIR}
  ${CMAKE_BINARY_DIR}
  ${KDE4_INCLUDES}
)
```

```
# We add our source code here
set(tutorial1_SRCS plasma-tutorial1.cpp)
```

```
# Now make sure all files get to the right place
kde4_add_plugin(plasma_applet_tutorial1 ${tutorial1_SRCS})
target_link_libraries(plasma_applet_tutorial1
  ${KDE4_PLASMA_LIBS} ${KDE4_KDEUI_LIBS})
```

```
install(TARGETS plasma_applet_tutorial1
  DESTINATION ${PLUGIN_INSTALL_DIR})
```

```
install(FILES plasma-applet-tutorial1.desktop
  DESTINATION ${SERVICES_INSTALL_DIR})
```

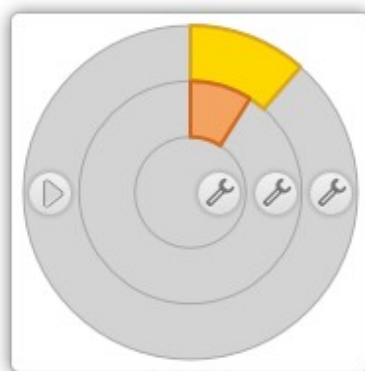
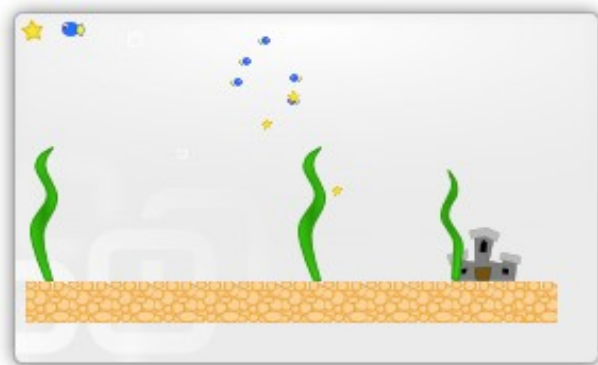
**Mettiamo insieme il tutto specificando  
nel file CMakeLists.txt  
le definizioni per la creazione del  
Makefile**

**Grazie al makefile generato  
potremo compilare la nostra applet  
digitando nella shell il comando  
'make'**

**L'installazione può essere  
eseguita in automatico  
digitando 'make install' da root \***

\* N.B. I percorsi specificati nel 'make install' possono variare a seconda della distribuzione usata

# Migliori widget



Acquarium

Make progress

ShortLog