OPTIMAL MIXING OF NATIVE FUNCTIONALITY AND WEB TECHNOLOGIES: CASE QT

Anna Szewczyk

Bachelor's Thesis
April 2011

Degree Programme in Information Technology
School of Technology

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Author(s) SZEWCZYK, Anna	Type of publication Bachelor´s Thesis	Date 02.04.2011
	Pages 45	Language English
	Confidential ( ) Until	Permission for web publication GRANTED
Title OPTIMAL MIXING OF NATIVE FUNCTIONALITY AND WEB TECHNOLOGIES: CASE QT		
Degree Programme Information Technology		
Tutor(s) PELTOMÄKI, Juha		
Assigned by Digia Finland Oyj		

Abstract

The main goal of the thesis assigned by Digia Finland Oyj was to explain what hybrid applications are, present the way of creating them and define the most optimal choices of their development. All these aspects are presented based on Qt, which provides an excellent set of tools for building hybrid applications.

With the internet and mobile industry development, end users' demands started to increase. Accessing the internet with common mobile browsers was not enough anymore. Following the customer's needs, developers all over the world had to investigate the problem of creating mobile applications, offering access to the most popular web services. This is how the idea of a hybrid application was raised. The assumption was to create availability to the service both for connected and disconnected users. The goal was to create a native application with no feature access restrictions, which is fast because demanding customers do not accept any delays and to attract as many potential users as possible the application had to be versatile.

The thesis describes the basic aspects of a hybrid application's idea. It presents to the readers simple examples of hybrid application building, how they look and what should be taken into account during the whole process of application development. In the thesis a way of combining the best parts of both native functionality and web technology are illustrated, which as a result gives a hybrid application - an application which can reign on the future mobile phones and tablets.

Keywords

Qt, QtWebKit, JavaScript, hybrid application, web application, native application.

Miscellaneous

CONTENTS

FIGURES

ABBREVIATIONS

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

BSD Berkeley Software Distribution license

CDB Microsoft Console Debugger

CSS Cascading Style Sheets

DRM Digital Rights Management

GDB GNU Symbolic Debugger

XHTML eXtensible HyperText Markup Language

XML Extensible Markup Language

XPath XML Path Language

HTML HyperText Markup Language

IDE Integrated Development Environment

IP Internet Protocol

LGPL GNU Library General Public License

NPAPI Netscape Plugin Application Programming Interface

QML Qt Meta-Object Language

RDF Resource Description Framework

SQL Structured Query Language

SSL Secure Socket Layer

SVG Scalable Vector Graphics

TCP Transmission Control Protocol

UI User interface

URL	Uniform Resource Locator
XAML	eXtensible Application Markup Language
XUL	XML User Interface Language

DEFINITIONS

Hybrid application – it is a combination of native and web application. The outer shell of the application is a native one and it contains web parts in form of primary user interface implemented in HTML, CSS and JavaScript and running inside of the platform's web control.

(Mills, 2009; Pavley, 2010)

Native application – it is an application designed to run on a device's operating system and machine firmware. Typically it needs to be adapted in order to be used on different devices.

(dotMobi, 2010)

Web application – it is an application which is accessed via network. It uses a web browser as a client, which means that browser is used as a program the person uses for running the application.

(Chaffee, 2000)

1 INTRODUCTION

Nowadays computer and internet are essential factors of our daily life. The power and range of the Internet has become so huge that almost no one imagines living without it. It is basic mass-media and people use it in all aspects of their life, starting from gaining new information from multimedia libraries, watching programs, finding places through online maps, striking up the acquaintances, sharing their photos etc.

Simultaneously with the development of the Internet, the demand for more and better applications supporting internet services has increased. Desktop applications were no more sufficient and that is how hybrid applications started to be developed.
With the development of ever fancier, faster and newer technologies meeting customer's expectations, the boundary between desktop applications and the web became very thin. Eventually these two distinct programming models were combined within one single application program creating hybrid application, which in the nearest future can reign the whole portable devices market.

Objectives of the thesis

The main goal of this Bachelor's thesis is to show what hybrid applications are, how they were formed and how they can be built. In order to do that firstly basic information about tools and technologies required for creating hybrid applications is presented and explained. There are many different technologies that can be used for creating a hybrid application, inter alia:

- OSGi + Java EE
- Flex + HTML
- Microsoft Silverlight + JavaScript
- Adobe AIR + HTML, JavaScript
- Appcelerator Titanium
- Phone Gap

- Oracle’s Java Platform integrated with Gecko
- Qt + WebKit.

However, due to the fact that currently Qt has become a more and more popular framework, it will be used in this document to show examples of hybrid application creation. That is why firstly the reader is familiarized with Qt and the possibilities and tools that it provides. Next the notion of WebKit and Qt WebKit are explained and their basic usage is shown. An inseparable part of hybrid applications is also JavaScript which will be described in one of the following chapters (3.2).

After characterizing essential technologies and tools needed for hybrid applications creation, the advantages and disadvantages of such an application are described. The basic aspect of creating hybrid application is presented - among other things, which parts of native functionality and web technologies should be used in order to create the best hybrid application are explained. Additionally, the author explains when it is worth to take the hybrid approach, and when it is better to choose a purely native or web application.

Finally, the problems that each hybrid application developer can face are demonstrated. The reader is also familiarized with the basic ways of solving such problems with the usage of Qt and the tools it provides as well as JavaScript. In the last paragraphs the author acquaints the reader with the basics of a QML language and presents an example of hybrid application.

2 QT FRAMEWORK

Qt is a set of cross-platform framework dedicated to C++. It was produced by the Norwegian company Trolltech and since 2008 it is owned by Nokia. The latest stable version 4.7.2 was released on the first of March 2011.

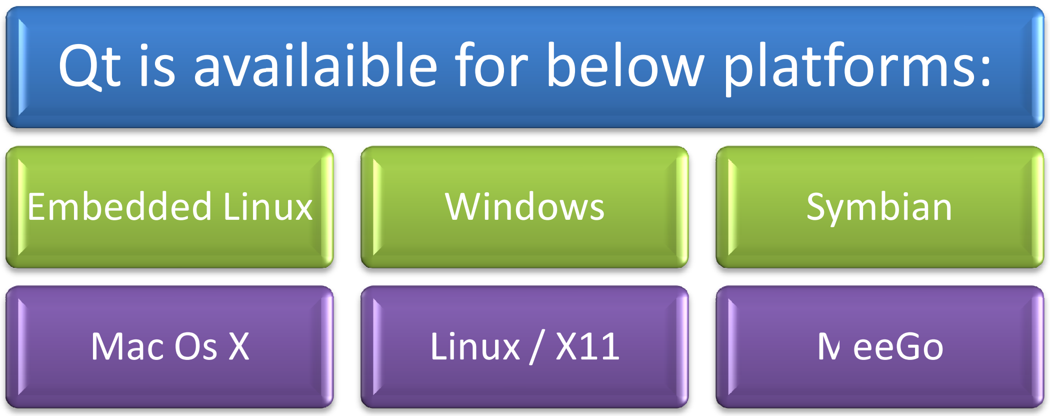


FIGURE 1. Availability of Qt framework on different platforms (Nokia, 2011)

Qt libraries were written in C++ and have fully object-oriented architecture. However, they can be also used, via language bindings, in applications written in several other languages, including: Java, C#, Ada, Pascal, Perl, PHP, Ruby and Python. Their main purpose is to create complex desktop applications with graphic interface. Qt libraries contain classes responsible for graphic elements and the ones responsible for lower level operations. Qt gives possibility of building multithread programs, working with files, sockets, signals and slots mechanism or hierarchical system for handling events. It also offers modules handling OpenGL, databases (SQL), XML localization and cross-compiling.

(Nokia, 2011)

Qt contains a number of specialized tools which can be divided into two groups: graphic and pre-compilation tools and they are presented below.

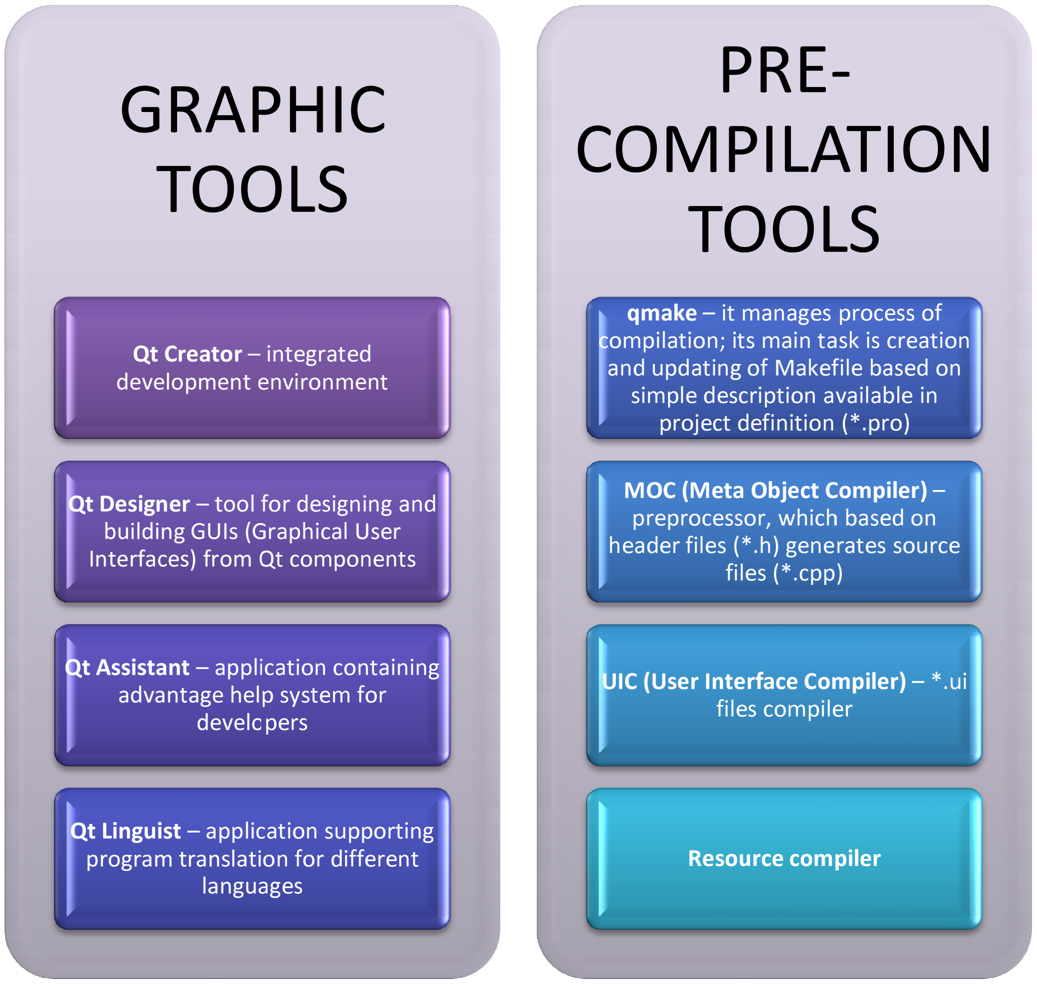

GRAPHIC TOOLS
Qt Creator – integrated development environment
Qt Designer – tool for designing and building GUIs (Graphical User Interfaces) from Qt components
Qt Assistant – application containing advantage help system for develcpers
Qt Linguist – application supporting program translation for different languages
PRE-COMPILATION TOOLS
qmake – it manages process of compilation; its main task is creation and updating of Makefile based on simple description available in project definition (*.pro)
MOC (Meta Object Compiler) – preprocessor, which based on header files (*.h) generates source files (*.cpp)
UIC (User Interface Compiler) – *.ui files compiler
Resource compiler

FIGURE 2. Graphic and pre-compilation tools available on Qt (Nokia, 2011)

The most important Qt Graphic tools are presented below as follows.

Qt Creator IDE

Qt Creator is freely available for download, cross-platform IDE, which enables creating applications for multiple desktop and mobile device platforms. The latest stable version 2.1.0 was released on the first of March 2011. It can be run on Windows, Linux/X11 and Mac OS X operating systems.

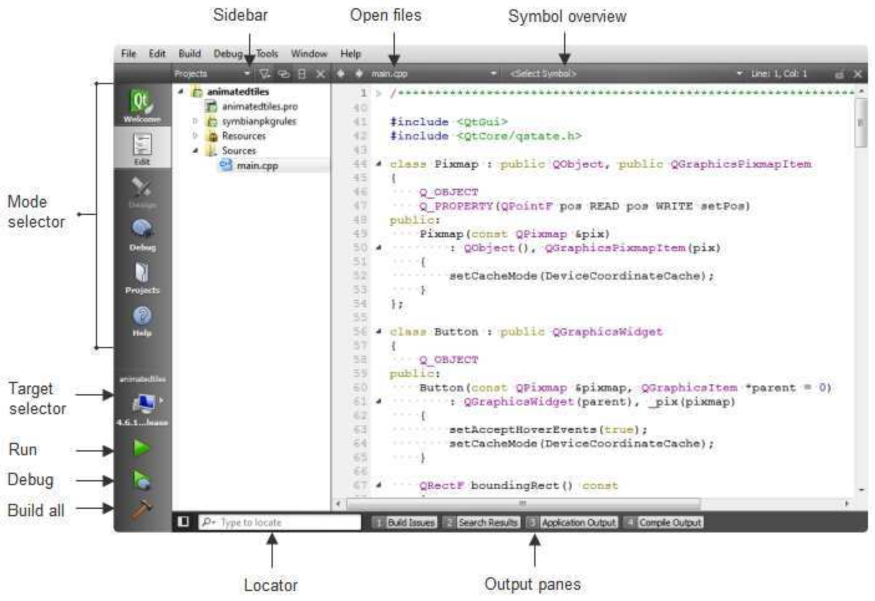
Qt Creator provides a set of very useful features, like C++ and JavaScript code editor. It supports different version control systems:

- CSV

- Git
- Mercurial
- Perforce
- Subversion

It also provides GDB, CDB and internal JavaScript debuggers, project and build management tools, simulator for mobile UIs and support for mobile and desktop targets. Qt Creator has also an integrated UI designer allowing modifying the UI files, which are XML files encoding the user interface in a machine-readable way.

(Nokia, 2011)

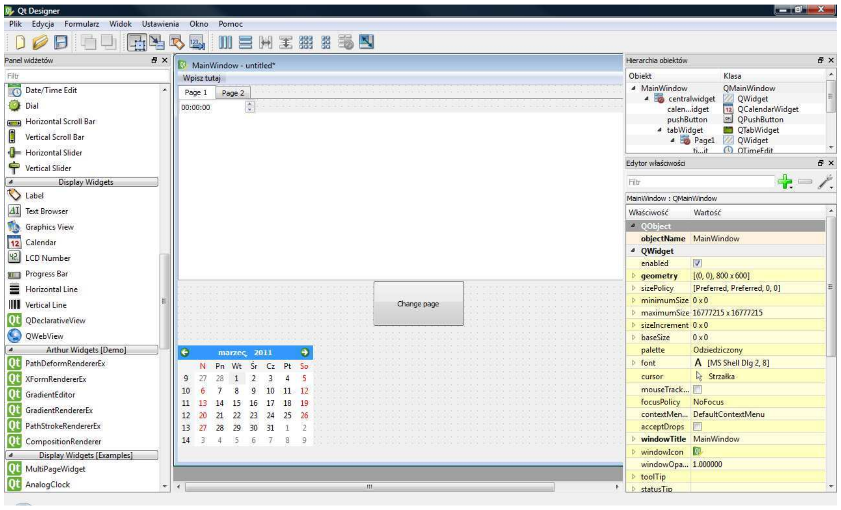


```
/*************************************************************

#include <QtGui>
#include <QtCore/qstate.h>

class Pixmap : public QObject, public QGraphicsPixmapItem
{
    Q_OBJECT
    Q_PROPERTY(QPointF pos READ pos WRITE setPos)
public:
    Pixmap(const QPixmap &pix)
        : QObject(), QGraphicsPixmapItem(pix)
    {
        setCacheMode(DeviceCoordinateCache);
    }
};

class Button : public QGraphicsWidget
{
    Q_OBJECT
public:
    Button(const QPixmap &pixmap, QGraphicsItem *parent = 0)
        : QGraphicsWidget(parent), _pix(pixmap)
    {
        setAcceptHoverEvents(true);
        setCacheMode(DeviceCoordinateCache);
    }

    QRectF boundingRect() const
```

FIGURE 3. Qt Creator in edit mode (Nokia, 2011)

Qt Designer

Qt Designer is an application for creating instantly previewed graphical interfaces with drag and drop functionality. It contains a library of standard widgets but enables also creating customized ones. Moreover it generates C++ or Java code directly from created interface. Qt Designer can be used with Visual Studio and Eclipse IDEs.



Obiekt	Klasa
MainWindow	QMainWindow
centralwidget	QWidget
calen...idget	QCalendarWidget
pushButton	QPushButton
tabWidget	QTabWidget
Page1	QWidget
ti...it	QTimeEdit

Właściwość	Wartość
QObject	
objectName	MainWindow
QWidget	
enabled	✓
geometry	[(0, 0), 800 x 600]
sizePolicy	[Preferred, Preferred, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Odziedziczony
font	A [MS Shell Dlg 2, 8]
cursor	Strzałka
mouseTrack...	☐
focusPolicy	NoFocus
contextMen...	DefaultContextMenu
acceptDrops	☐
windowTitle	MainWindow
windowIcon	
windowOpa...	1.000000
toolTip	
statusTip	

FIGURE 4. Qt Designer in edit mode (Nokia, 2011)

Currently Qt is very popular and it is used, inter alia, to develop the following apps:

- Adobe Photoshop Album
- Gadu-Gadu (v8)
- Google Earth
- Mathematica
- Opera
- Opie
- Picasa
- Skype
- VLC media player
- VirtualBox

(Ovisoftware, 2010)

It is also used as development tool, inter alia, by following companies:

- Autodesk
- European Space Agency
- KDE

- Panasonic
- Philips
- Samsung
- Siemens
- Volvo
- Walt Disney Animation Studios

(Panayara, 2011)

The constantly growing popularity of Qt, its versatility in terms of its possible use on various platforms and also whole bunch of tools provided by it, decided on its choice as an exemplary framework for showing hybrid application creation. Furthermore, Qt has a rich selection of features for hybrid apps development: Qt WebKit, QtScriptModule, QtXMLPatterns and QNetworkAccessManager.

(Nokia, 2011)

Qt & Digia company

Digia Plc as one of the world's largest independent provider of OS based software integration services and solutions for smartphones with a unique expertise in, inter alia, mobile devices and web application design, has been involved in Qt development for a long time. It has not only employed many Qt developers, prepared Qt trainings, taken part in different Qt events, for which it has always prepared varied showcases but it has also developed a many QT applications. One of them was released in the year 2009: Digia @Web, which is fully finger touch controllable, web browser specifically designed application to run on Maemo (FIGURE 5).

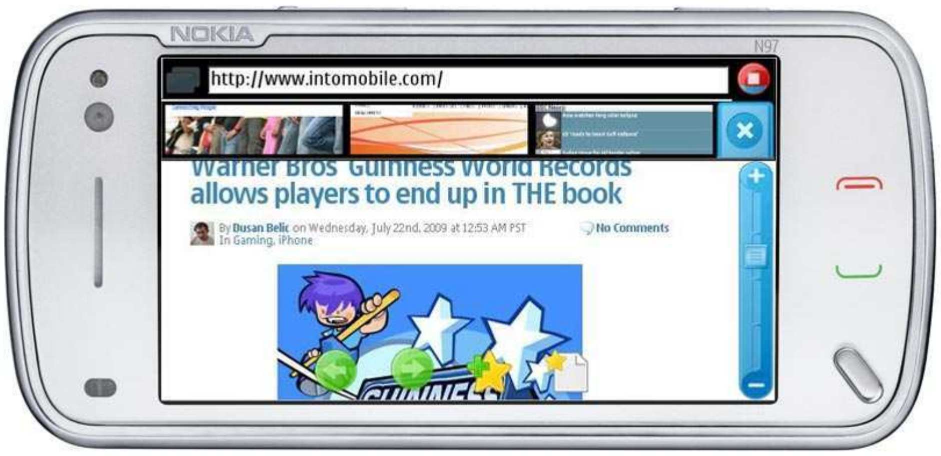

NOKIA
N97
http://www.intomobile.com/
Warner Bros' Guinness World Records allows players to end up in THE book
By Dusan Belic on Wednesday, July 22nd, 2009 at 12:53 AM PST In Gaming, iPhone
No Comments

FIGURE 5. Digia @Web Qt browser (Digia Plc, 2011)

Currently Digia has approached even closer to the Qt, due to the fact that on the seventh of March 2011 it has signed the agreement with Nokia to acquire the Qt commercial licensing and service business. It is expected that Digia will be involved in further Qt growth even more than up to present time. With hundreds of Qt experts it will contribute to Qt innovations and support further Qt growth.

(Digia Plc, 2011)

3 Qt WEBKIT

3.1 WebKit

WebKit is a browser engine developed as open source software. It allows rendering internet pages, HTML and executing JavaScript. Originally it was authored in C++ and used in Apple’s web browser – Safari. Later it has been ported to many frameworks. Currently WebKit is used in many applications: browsers, E-Mail Clients, Instant Messenger Clients, Web Development Applications etc. It is a technology which not only web developers but also end users are familiar with and it has been also widely used on mobile devices.

WebKit contains two of the most important libraries: WebCore and JavaScriptCore, which are available as system frameworks. WebCore and JavaScriptCore are developed on LGPL license and the other WebKit components are available on BSD license.

WebCore is a layout allowing processing and rendering web pages. It is Document Object Model library for HTML and SVG that is a way of presenting complex XML and HTML documents as an object model. This model is programming language and platform independent.

JavaScript Core is a framework implementing usage of JavaScript language, which is described more precisely in the next subsection.

(The WebKit Open Source Project, 2011)

3.2 JavaScript

JavaScript was originated in Netscape company as LiveScript –scripting language expanding HTML. In 1995, based on the agreement between Netscape and Sun Microsystems companies a new language – JavaScript – was created.

It is an implementation of the ECMAScript language standard, which is widely used for client-side scripting on the web. JavaScript is scripting, object-oriented, interpreted language, which allows creating short programs directly embedded in HTML code. These short JavaScript parts can recognize and properly react on different events caused by end user.

JavaScript is a multiplatform language, which means that the code written in JavaScript, in most cases, can be used on different operating systems (like Windows, Linux, MacOS). What is more, on all of these systems the scripts should give the same results. It is very important because currently computers and other devices - used for Internet, browsing, are created on various operating systems. Without the cross-platform nature of JavaScript and its wide range of properties, developing scripts improving WWW pages for users working on different kind of systems, would be much more difficult. Moreover, the cross-platform property of JavaScript -

contributed to its increasing popularity, because architects of web pages noticed a chance in this language to reach the widest group of end users.

JavaScript’s main application is to improve web pages. It is done by introducing dynamic and interactive effects to static pages, which is impossible to be obtained only with usage of HTML. It can be used for implementing error handling for data entered to HTML forms. JavaScript can also be used to work with XML data as well as with XML based languages such as XHTML, SVG, XAML, RDF, XUL etc.

The main JavaScript features are:

- cross-platform, lightweight, object-oriented, scripting, interpreted language
- it is based on predefined objects but it does not allow to use all of the object-oriented programming mechanisms like inheritance
- it can react to different kind of end user’s events, like mouse clicking, navigating between pages etc.
- it is embedded in HTML code
- it introduces dynamic and interactive effects to the static web pages
- it can handle errors caused for example by data entered to HTML forms
- it can create cookies on browser side.

(Watt, Watt, Simon, & ODonnell, 2003, 21-29)

3.3 Qt WebKit

Qt WebKit is a Web browser engine, based on the Open Source WebKit engine, integrated with Qt. As it is a web contents rendering engine, it is also one of the most important components required for creating hybrid applications. With this module it is much easier to integrate Open Source WebKit into desktop applications. It is fully featured and standard compliant. It supports JavaScript, HTML / XHTML, CSS, SVG, XPath and AJAX technologies. This is a very useful module and there are already many applications using it, for example:

- Arora
- Konqueror
- Rekonq
- Light
- Orange Mobile Browser
- Google Earth
- Spotify for Linux
- grr
- Crochik Macuco
- qreddit
- Qt WRT.

(Nokia, 2011)

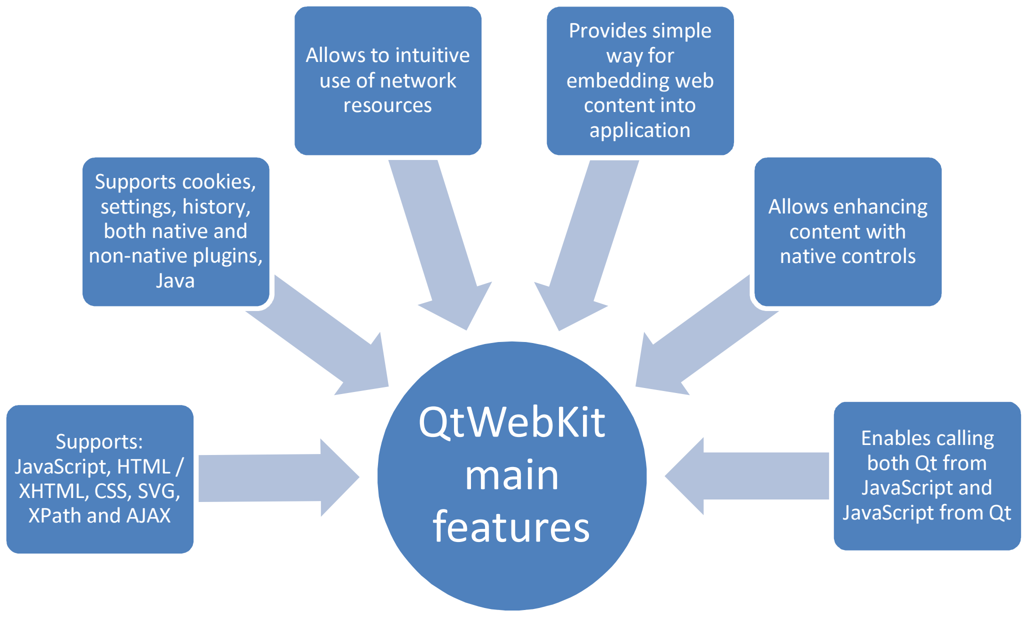

Allows to intuitive use of network resources
Provides simple way for embedding web content into application
Supports cookies, settings, history, both native and non-native plugins, Java
Allows enhancing content with native controls
Supports: JavaScript, HTML / XHTML, CSS, SVG, XPath and AJAX
QtWebKit main features
Enables calling both Qt from JavaScript and JavaScript from Qt

FIGURE 6. Qt WebKit main features (Nokia, 2009) (Kosonen, 2009)

Qt WebKit has native application integration which means that local and web content can be merged. It is a basic module used for creating hybrid application, mainly because it provides components which allow integration of web technologies. It has also varied set of tools and features, some of which are described in FIGURE 6 and it also has modern rendering features presented in FIGURE 7.

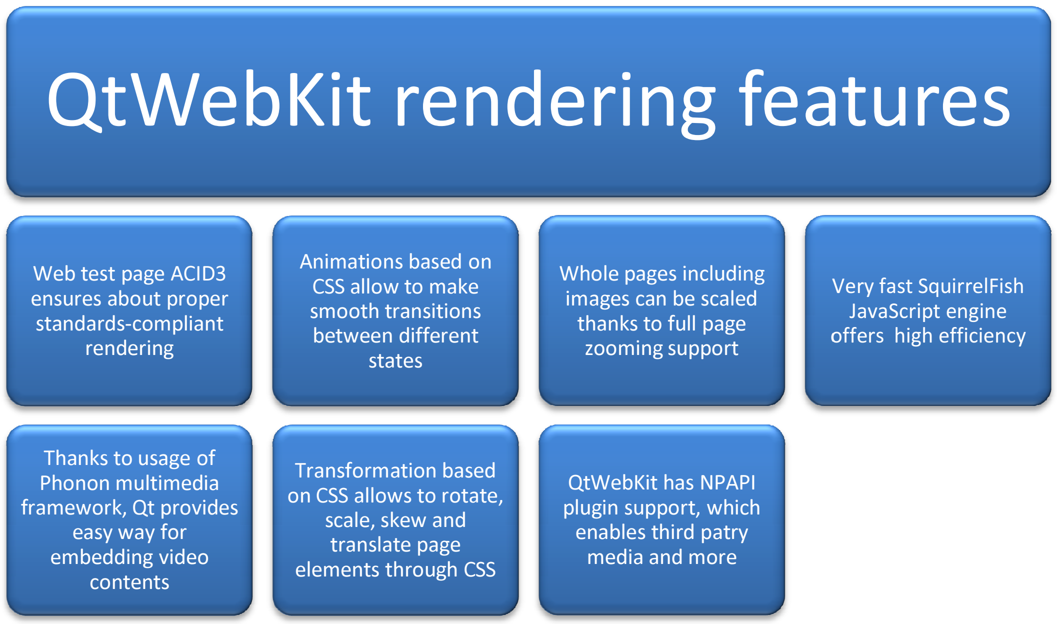

QtWebKit rendering features
Web test page ACID3 ensures about proper standards-compliant rendering
Animations based on CSS allow to make smooth transitions between different states
Whole pages including images can be scaled thanks to full page zooming support
Very fast SquirrelFish JavaScript engine offers high efficiency
Thanks to usage of Phonon multimedia framework, Qt provides easy way for embedding video contents
Transformation based on CSS allows to rotate, scale, skew and translate page elements through CSS
QtWebKit has NPAPI plugin support, which enables third patry media and more

FIGURE 7. Qt WebKit rendering features (Nokia, 2009)

3.4 Qt WebKit classes

Qt WebKit module provides a great number of classes which allow rendering HTML, XHTML and SVG documents. The following classes can be found in Qt WebKit:

- QWebFrame – it is responsible for frame inside a web page
- QWebHistory – it is directly related to the history of visited pages. Thanks to this class it is possible to navigate history
- QWebHistoryInterface – this class gives an interface, which is used to create link history

- QWebHistoryItem – as the name suggests, this class is simply responsible for one single item in the history. QWebHistoryItem stores such as properties of the web page like: title, URL, date and time of the last visit on the page and an icon of the page
- QWebHitTestResult – this class returns web page content's information after hit test
- QWebPage – it allows viewing and editing web page. It also holds a main frame which is directly related to web content, its whole history and settings
- QWebPluginFactory – it is responsible for creating plugins
- QWebSettings – this class keeps all settings configured for the page
- QWebView – it allows rendering web page content. It consists of the objects of the other Qt WebKit classes like QWebFrame and QWebPage.

(Nokia, 2011)

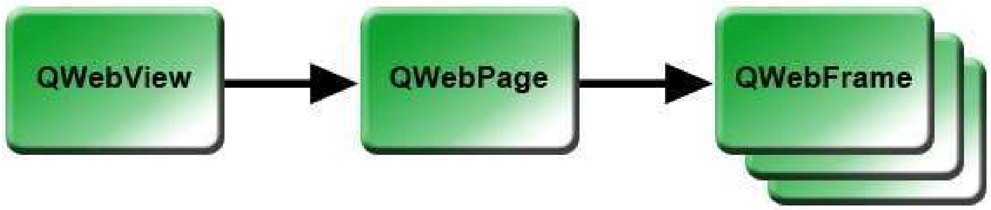


FIGURE 8. Qt WebKit architecture (Nokia, 2011)

The list above presents all classes incorporated into the Qt WebKit module. There are, however, also other TCP / IP protocol based classes related to this module that were introduced in the Qt to provide a complete browser framework.

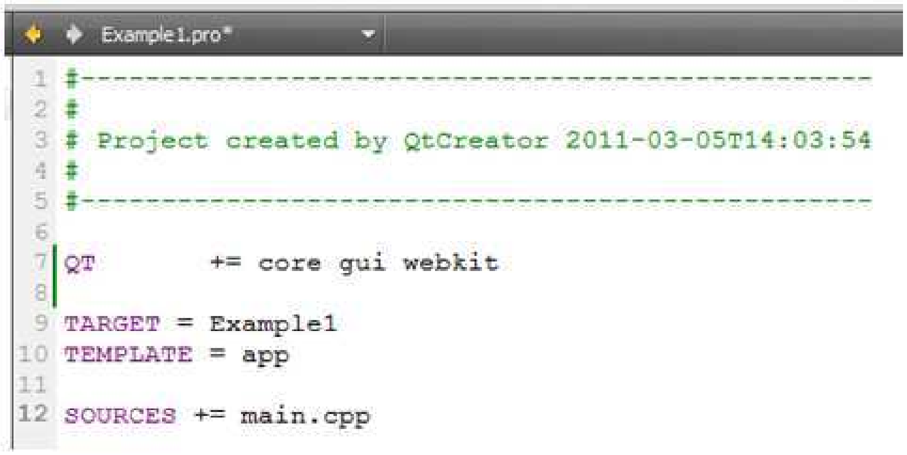
- QNetworkAccessManager – providing functionality for sending network requests and receiving replies

- QNetworkRequest and QNetworkReply- they hold data related to request sent and reply received with QNetworkAccessManager
- QNetworkCookie – it stores one network cookie
- QNetworkCookieJar – it stores all cookies (QNetworkCookie) from previous request
- QAuthenticator – when accessing service requires authentication, this class allows to give authentication information to the socket
- QTcpSocket and QSslSocket – they provide TCP and SSL sockets

(Nokia, 2011)

3.5 Qt WebKit simple example

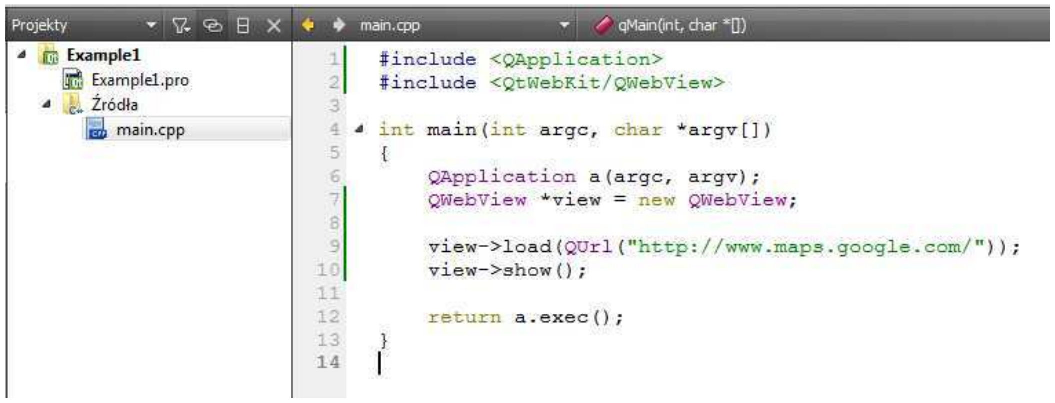
The simplest Qt WebKit example is an application based on QWebView class, which provides a widget for viewing and editing web documents. To start using Qt WebKit we need to add it to the project file (.pro file) of our application, as shown in FIGURE 9 in seventh line.



```
 1 #-------------------------------------------------
 2 #
 3 # Project created by QtCreator 2011-03-05T14:03:54
 4 #
 5 #-------------------------------------------------
 6
 7 QT       += core gui webkit
 8
 9 TARGET = Example1
10 TEMPLATE = app
11
12 SOURCES += main.cpp
```

FIGURE 9. Example of project file containing webkit

After that we can start creating the first program. Firstly we need to create a new QWebView widget (FIGURE 10 line 7). With the load() function (FIGURE 10 line 9) an exemplary web page will be loaded ("http://www.maps.google.com/"). onto QWebView widget. To display QWebView widget, show() function has to be invoked (FIGURE 10 line 10).



```
1  #include <QApplication>
2  #include <QtWebKit/QWebView>
3
4  int main(int argc, char *argv[])
5  {
6      QApplication a(argc, argv);
7      QWebView *view = new QWebView;
8
9      view->load(QUrl("http://www.maps.google.com/"));
10     view->show();
11
12     return a.exec();
13 }
14
```

FIGURE 10. Example of Simple Qt WebKit application

After building and running this example we will be able to use http://www.maps.google.com/ web page – search different places, zoom in and zoom out the map, pan the map etc. But there will be no possibility of using more advanced functionality that is normally provided by each browser, like going back or forward, changing web page etc. FIGURE 11 shows examples of how this application works and what its limitations are.

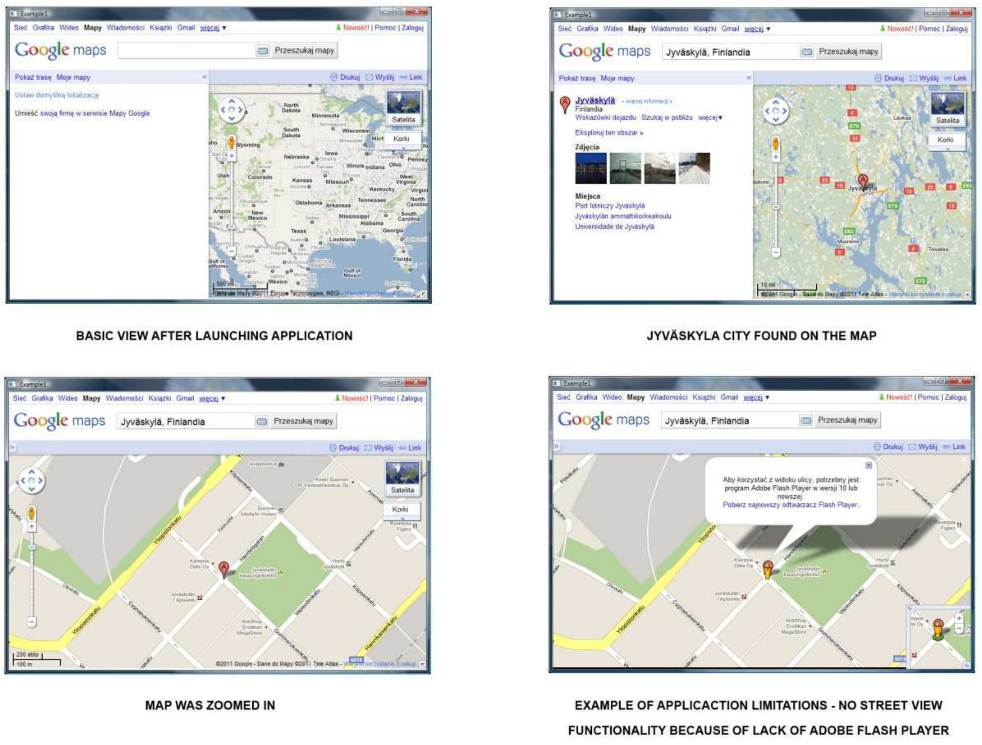

BASIC VIEW AFTER LAUNCHING APPLICATION
JYVÄSKYLA CITY FOUND ON THE MAP
MAP WAS ZOOMED IN
EXAMPLE OF APPLICACTION LIMITATIONS - NO STREET VIEW FUNCTIONALITY BECAUSE OF LACK OF ADOBE FLASH PLAYER

FIGURE 11. Application performance and its limitations: examples

4 HYBRID APPLICATION

4.1 Brief definition of hybrid application

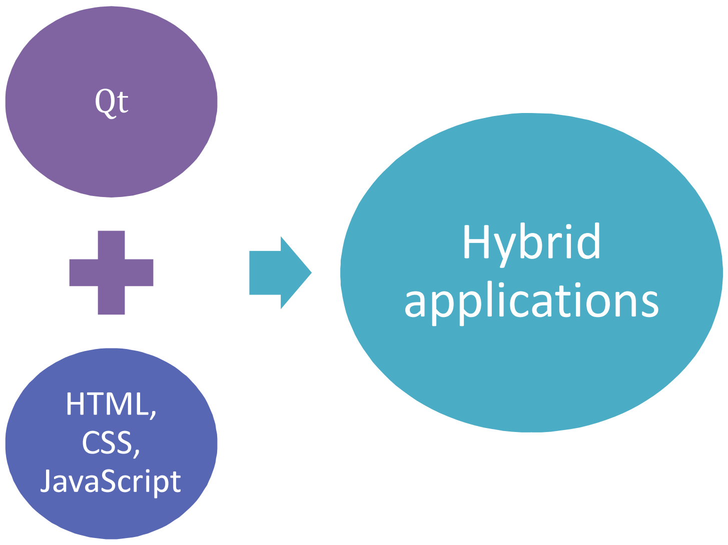


FIGURE 12. Hybrid application as combinations of native and web technologies

Hybrid application is more of a program than a web site, because it has the ability to work in offline mode. It uses both features of native and web technologies as well as it has some of their properties. In general, a hybrid application is somewhere between web and native applications trying to get as much from both of them as possible - it tries to be as reachable as a web app and as rich as a native app.

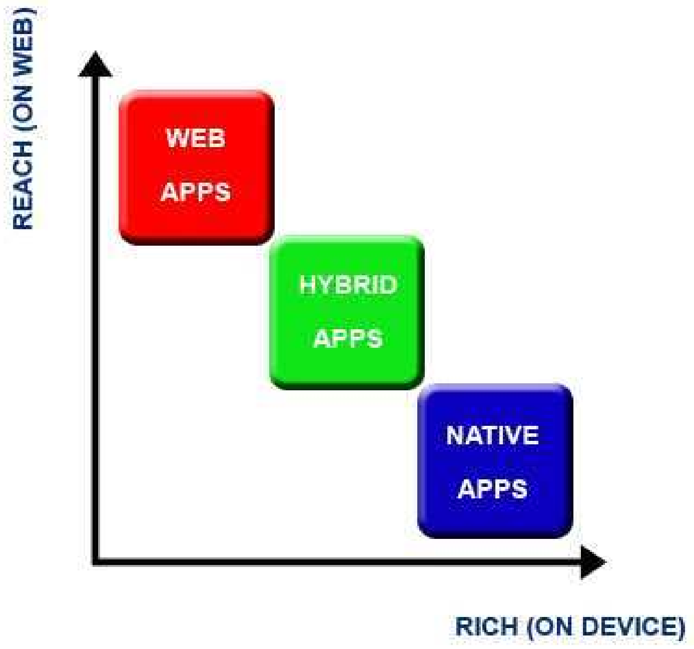


FIGURE 13. Hybrid application in relation with Web and Native applications (Nokia, 2010)

4.2 Benefits of hybrid, native and web technologies

As already mentioned, hybrid applications are a combination of native and web technologies and they mix the best features of both of them. However, hybrid applications cannot completely replace native or web applications. There are still situations when it is more beneficial to use original technologies instead of their combination, that is, the hybrid application. The figure below shows when each type of application should be chosen, in order to create the program that will meet all expectations of both developers and end users.

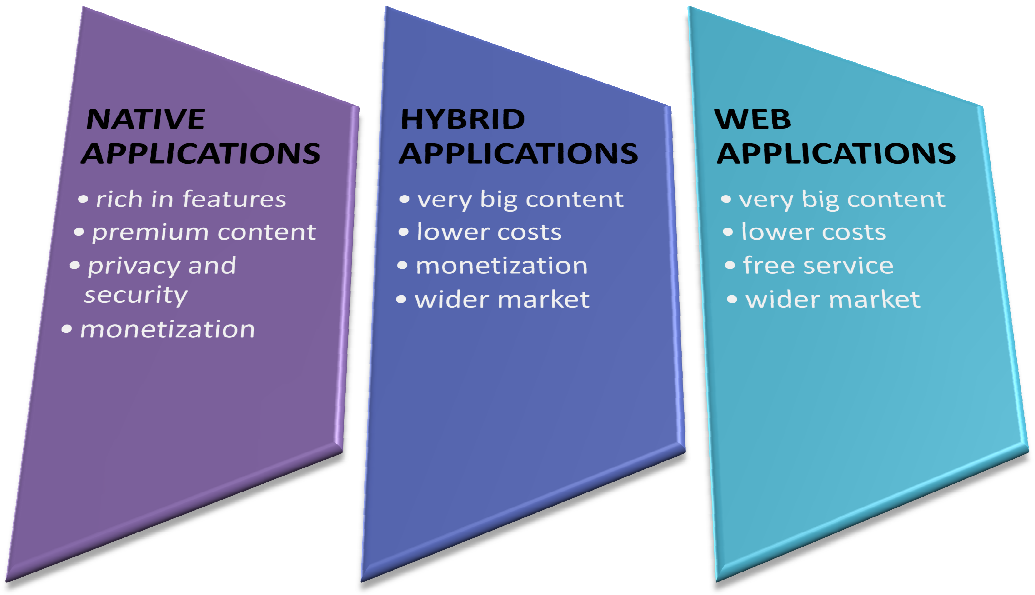


FIGURE 14. Image showing when each type of application should be used (Nokia, 2010)

From the figure above it is clearly noticeable that again the hybrid application has features of the two other technologies. But in a situation when the type of technology needs to be chosen to meet some predefined criteria each of the application will be suitable for different purposes.

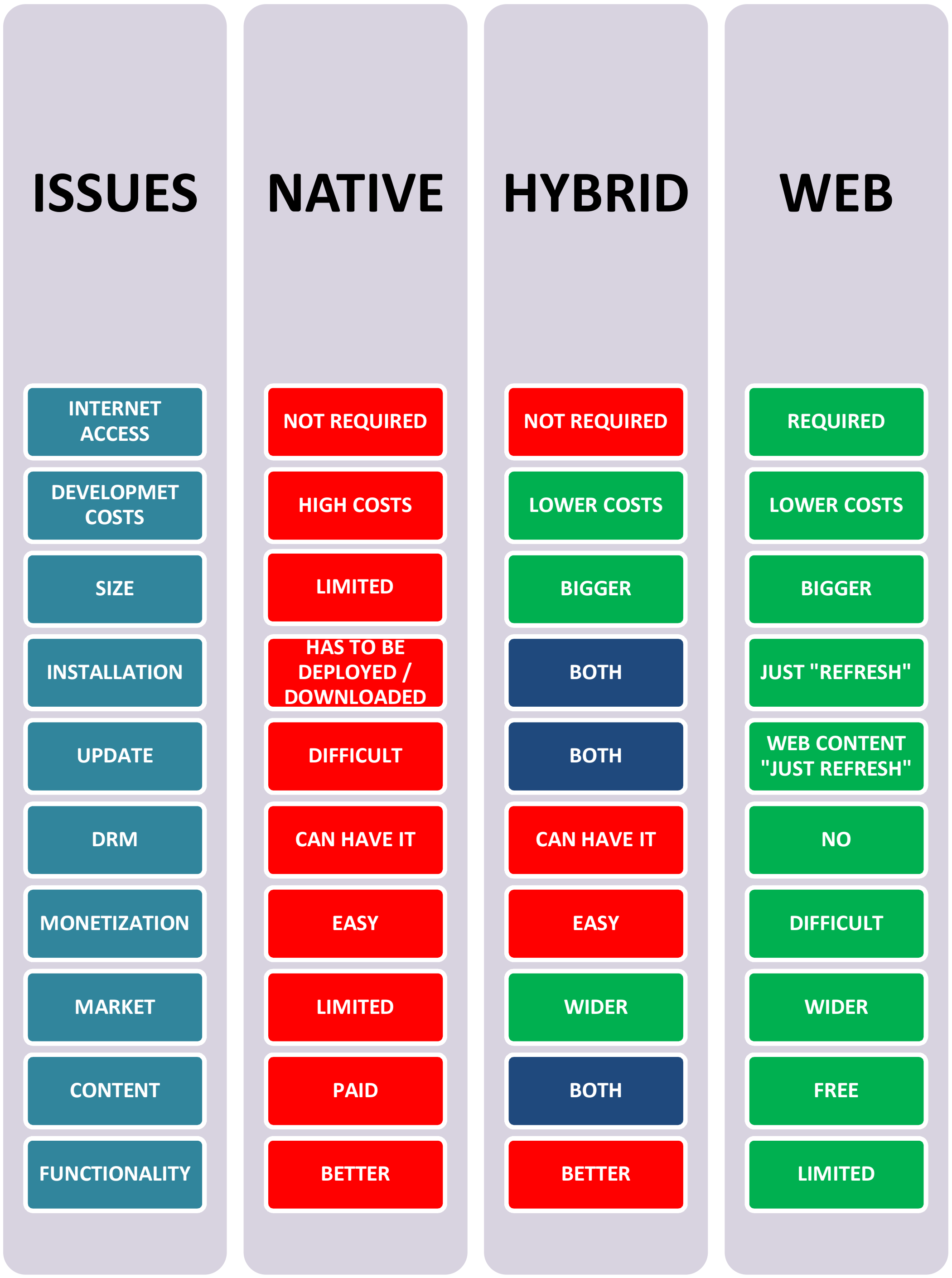
Native app is definitely used when a developer wants to make money on the product which has enhanced functionality, premium content and is secure and private. Good examples of such applications are different kind of unique games where no developer wants to share their content for free or in a way that are used as part of some other program.

Web applications should be used whenever the content is heavy and requires storing on servers. These kinds of apps are usually free services and most often if their owners make some money on these kinds of applications, it is not done directly on them but for example on different kind of advertisements located in apps. Web applications are also very popular because they use web browser as a client, which is not only ubiquitous but also in most cases simple to use. Exemplary web apps can be Social Networks or bank applications.

In turn, hybrid applications - the same as native ones - are easy to be monetized but also the same as web ones they have wider potential market and can have big content.. This group may include, for example, some maps or a specialized medical application.

It is more clear now that each type of application is more suitable depending on situation and its purpose. FIGURE 14 shows also some common features for native and hybrid apps or web and hybrid ones. This is, however, only a small part of the large number of properties which connect these technologies. FIGURE 15 shows a better comparison between all of them and indicates which features of hybrid apps were taken from native and web technologies.

(Nokia, 2010)



ISSUES	NATIVE	HYBRID	WEB
INTERNET ACCESS	NOT REQUIRED	NOT REQUIRED	REQUIRED
DEVELOPMET COSTS	HIGH COSTS	LOWER COSTS	LOWER COSTS
SIZE	LIMITED	BIGGER	BIGGER
INSTALLATION	HAS TO BE DEPLOYED / DOWNLOADED	BOTH	JUST "REFRESH"
UPDATE	DIFFICULT	BOTH	WEB CONTENT "JUST REFRESH"
DRM	CAN HAVE IT	CAN HAVE IT	NO
MONETIZATION	EASY	EASY	DIFFICULT
MARKET	LIMITED	WIDER	WIDER
CONTENT	PAID	BOTH	FREE
FUNCTIONALITY	BETTER	BETTER	LIMITED

FIGURE 15. Comparison of attributes taken from Native and Web applications (Nokia, 2010)

As presented in the FIGURE 15 hybrid applications mix the best attributes of native and web apps. First of all, the same as a native app, the hybrid one does not require constant internet access to be used. Of course it utilizes web services but it should be created in a way allowing for offline usage. A significant benefit from native technology is also better functionality which allows creating richer application if only because it helps to improve the web part of the hybrid application. Hybrid applications have also better security comparing to web ones (for example they can have DRM) and they are easy to be monetized.

There are, however, also many benefits that hybrid applications “inherit” from web technologies. First of all, they can be larger than normal native apps. This is because the web part of the hybrid app is not stored on the potential device but on different servers. Directly related to that are lower costs of application development since part of it is still web service. Hybrid applications the same as web ones have a wider market. As mentioned previously usually they use a common web service as a part of the program and they are created to be used not only by a limited group of customers but all of the device’s users.

FIGURE 15 shows also that there are attributes taken partly from native and partly from web functionalities. This group includes installation and update which of course needs to be done like for normal desktop applications, that is the whole program needs to be deployed and downloaded, on the other hand, the web section is regularly updated through the internet. The same situation is with its content – potential customer needs to pay for the application, however, the whole web content is free.

(Nokia, 2010)

5 Qt WEBKIT INTEGRATION

Due to the fact that hybrid applications are simply a mix of both native and web functionalities, it is worth understanding exactly which features of these technologies form a hybrid application.

If we consider Qt as native part, it is easy to extract elements that can be used from this framework, first of all, C++ as development language, but also a large set of reusable libraries, a rich set of widgets and what is very important, easy access to device API. On the other hand, the web part supplies the hybrid application in: JavaScript (and also JavaScript libraries), HTML and CSS but also more rapid development.

It is clearly visible that both native and web technologies give a varied set of useful tools that can be used to create very strong application. Of course with all these tools come not only their advantages but also disadvantages.

(Lal, 2009)

5.1 Hybrid application – basic problems

While creating different kinds of applications, the developer faces various problems. They depend on the type of applications, tools and technologies that are used etc.. The same situation takes place when hybrid applications are implemented. Of course some of the issues are directly related to the application content but there are also some that are quite common for all hybrid applications. Such problems can include, inter alia:

- interaction between native and web content
- access to data
- sudden network lost and offline mode
- web data processing

There are different kinds of solutions that can be used to handle each of these problems. Everything depends on specific technologies that are used. The next subsections show what kind of methods can be used to handle the above issues in the case of the use of Qt as native functionality.

5.2 Interaction between native application and rendered content

Hybrid application is a combination of both native and web technologies. In order to be used together, however, there needs to be some interaction between both of these functionalities. For this purpose Qt WebKit was introduced, which provides a set of features allowing collaboration between native application and rendered content, as it is shown in FIGURE 16 and described in the following subsections.

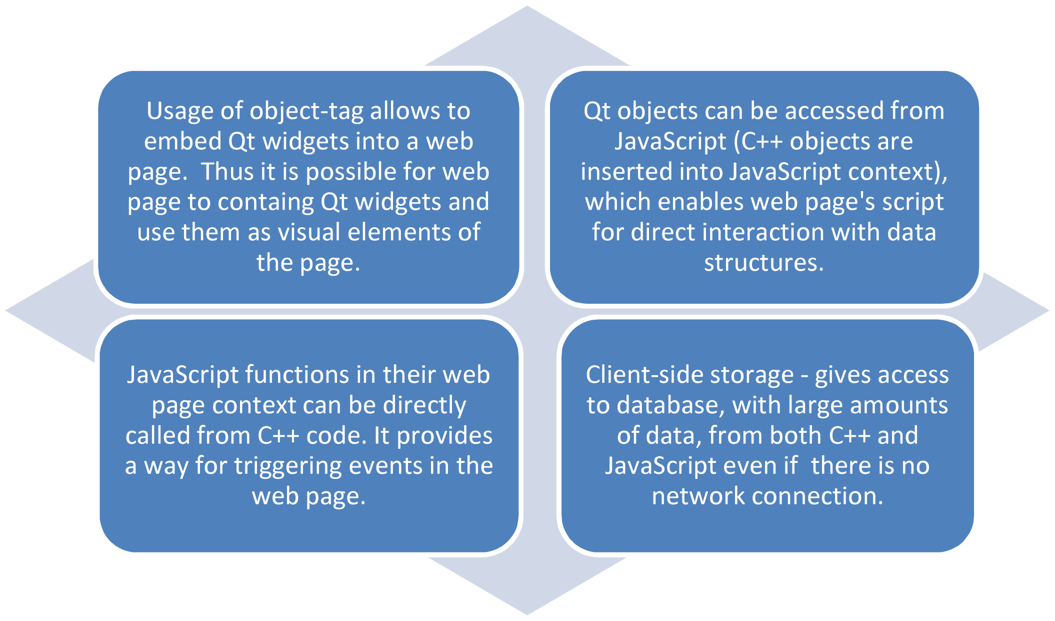

Usage of object-tag allows to embed Qt widgets into a web page. Thus it is possible for web page to containg Qt widgets and use them as visual elements of the page.
Qt objects can be accessed from JavaScript (C++ objects are inserted into JavaScript context), which enables web page's script for direct interaction with data structures.
JavaScript functions in their web page context can be directly called from C++ code. It provides a way for triggering events in the web page.
Client-side storage - gives access to database, with large amounts of data, from both C++ and JavaScript even if there is no network connection.

FIGURE 16. The ways of interactions of native and web functionalities (Nokia, 2009)

5.3 Embedded Qt Objects

The Qt WebKit provides functionality for efficient integration of native C++ and JavaScript. Two different ways of embedding Qt objects into a web page are described below:

- Writing browser plug-ins which enables embedding Qt objects into a web page using object-tag. An exemplary way of writing such a simple plug-in and its usage are shown below:

```
QObject* MainWindow::createPlugin(const QString &classid)
{
   QWidget *widget = 0;
   if(classid == "MyContacts")
   {
       widget = new MyContacts;
       return widget;
   }
}
```

The fragment of code above shows the native functionality which can be used with object-tag:

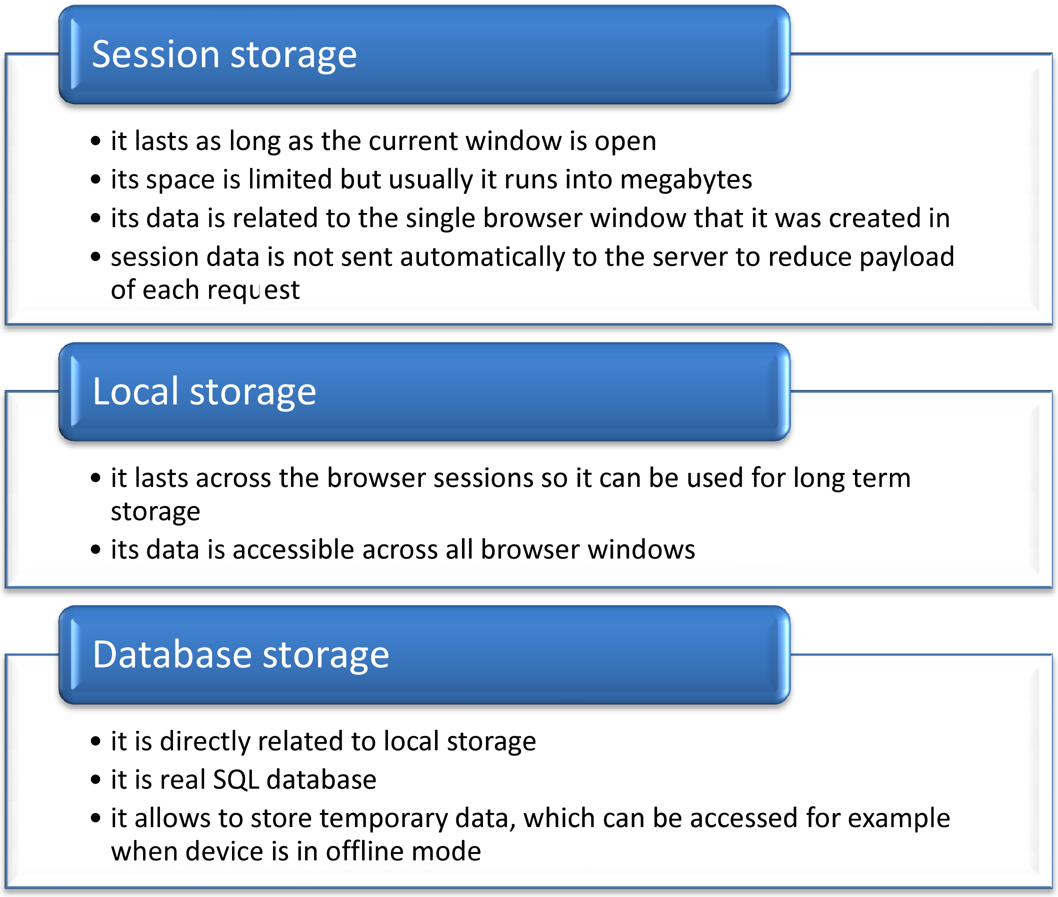
```
<OBJECT type="application/x-qt-plugin" classid="MyContacts"/>
```

- Adding Qt objects directly to the JavaScript context of the page can be done using the method:
  `QWebFrame::addToJavaScriptWindowObject`
  The above function does not ensure that after loading a new URL Qt object remains accessible. To do that Qt object needs to be added to the slot connected to the signal below:
  `QWebFrame::javaScriptWindowObjectCleared`

(Nokia, 2009)

5.4 Client-side storage

If the software application has to be useful, its data have to be stored in a persistent way. Hybrid application has the same issue to be solved but with the HTML5 advent, the new way of storing data appears, namely client-side database storage or “offline storage”. Client-side storage is divided into three methodologies, which are described in FIGURE 17.



Session storage

- it lasts as long as the current window is open
- its space is limited but usually it runs into megabytes
- its data is related to the single browser window that it was created in
- session data is not sent automatically to the server to reduce payload of each request

Local storage

- it lasts across the browser sessions so it can be used for long term storage
- its data is accessible across all browser windows

Database storage

- it is directly related to local storage
- it is real SQL database
- it allows to store temporary data, which can be accessed for example when device is in offline mode

FIGURE 17. Client-side storage methodologies (Dhandhania, 2011)

Based on FIGURE 17, client-side storage enables storing at least some temporary data when the device is offline. This is very important for hybrid technology because it allows to reduce traffic and also to use the application even when there is no network connection. Of course, it can be used for storing large quantities of data in a structured way, however, each developer should figure out themselves what data are really needed and should be stored in “offline database”.

The database can be accessed in two ways: either from JavaScript or through Qt’s SQL module using Qt WebKit.

The first way consists of searching database from JavaScript code and based on the results generating HTML. Below there is an exemplary code, (the part about creating database is omitted) showing the main idea of this method. It uses two main functions

- `openDatabase` – it opens a database if it exists, if not a new one is created and opened

- `transaction(function(tx)}{});` - it runs a database transaction and what is important it has the ability to roll back changes.

The example below opens an already existing database “CS_db” and runs its transaction. It selects the content of “userId” and “userText” from “cs” database, process text of returned results and in case of any errors it displays a proper error message.(Nokia, 2009)

```
<html>
  <head>
      <script language ="javascript">
      ...
      ...
      db = openDatabase("CS_db", "1.0", "Client-side storage 
database", 200000);
      db.transaction(function(tx)
      {
            tx.executeSql("SELECT userId, userText FROM cs",[],
function(tx, result)
            {
                  var len = result.rows.length, i;
                  for (i = 0; i < len; i++)
                  {
                        var row = result.rows.item(i);
                        processText( row['userId'], row['userText']
);
                  }
            },
            function(tx, error)
            {
                  alert('Error processing SQL - ' + error.message);
                  return;
            });
      });
      </script>
  </head>
  <body onload="onBodyLoad()">
    <h1>Client-side storage database example</h1>
  </body>
</html>
```

(Sharp, 2010)

The second method, using Qt WebKit, allows to access database through Qt's SQL module. From hybrid application point of view it is a very valuable feature. One of the reasons is that for example the web part of the application can use the same mechanism for storing data as when sharing them with the native part of the application. What is very important is that the client side database can be accessed only through JavaScript from the right security origin. It has to be done in this way to avoid any kind of security breaches. To access all security origins from C++ code there are two methods which can be used: `QWebSecurityOrigin::allOrigins()` and `QWebFrame::securityOrigin()`.

The code below shows how to access HTML5 database created with JavaScript. It can be done using `QWebSecurityOrigin::databases()` function, which returns the database defined by security origin. Then a new object of `QSqlDatabase` class – `sqldb` is created, which allows access to the database through connection. Using `QSqlDatabase::addDatabase()` function a database connection with driver type `"QSQLITE"` and the connection name `"webconnection"` is added to the previously created `QSqlDatabase` object. Afterwards using `fileName()` method, the name of the web database is returned and assigned to `sqldb` database with the usage of `setDatabaseName()` function. To have an effect, the connection's database name needs to be assigned before the connection is open, which is done in the last part of the code.

```
QWebDatabase webdb = mySecurityOrigin.databases()[index];
QSqlDatabase sqldb = QSqlDatabase::addDatabase("QSQLITE",
"webconnection");
sqldb.setDatabaseName(webdb.fileName());
if (sqldb.open())
{
    QStringList tables = sqldb.tables();
    ...
}
```

(Nokia, 2009)

5.5 Web data processing

When creating hybrid applications, there is still a problem of presenting data available through the web. Some of the specific data formats (geo data, news feeds etc.) cannot be directly presented in the program, but Qt has also solved this problem by preparing Qt's networking module allowing to download data in an easy way. Further data processing can be handled either by custom code or for example if the output format is XML or it should be displayed in the web page XHTML, and then it can be done by the QtXmlPatterns module.

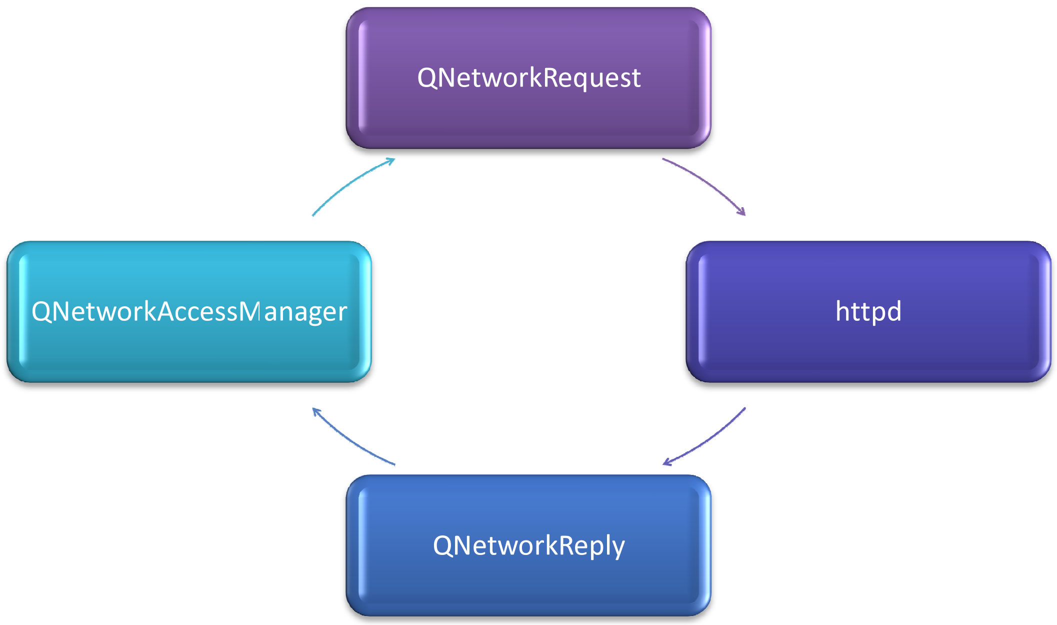


FIGURE 18. Network Access (Johnson, 2008)

Downloading the data from the web server using Qt's networking module is simple and this is presented in FIGURE 18. QNetworkAccessManager creates QNetworkRequest, which is sent to http web server. The web server returns a network reply which is encapsulated in QNetworkReply object. The example below shows how it is done in practice. `QNetworkAccessManager::get()` method is called to obtain the reply to the sent request. The result is received through signal: `finished(QNetworkReply*)` and there is also a possibility to monitor the progress of request by listening to the signal: `downloadProgress(qint64, qint64).`

```
QNetworkAccessManager *manager = new QNetworkAccessManager(this);
connect( manager, SIGNAL(finished(QNetworkReply*)),
          this, SLOT(replyFinished(QNetworkReply*)));
connect( manager, SIGNAL(finished(QNetworkReply*)),
        m_progressBar, SLOT(hide()) );
QNetworkReply *reply = manager->get( QNetworkRequest( QUrl(Url ) ) );
connect( reply, SIGNAL(downloadProgress(qint64, qint64)),
this, SLOT(updateProgress(qint64,qint64)) );
```

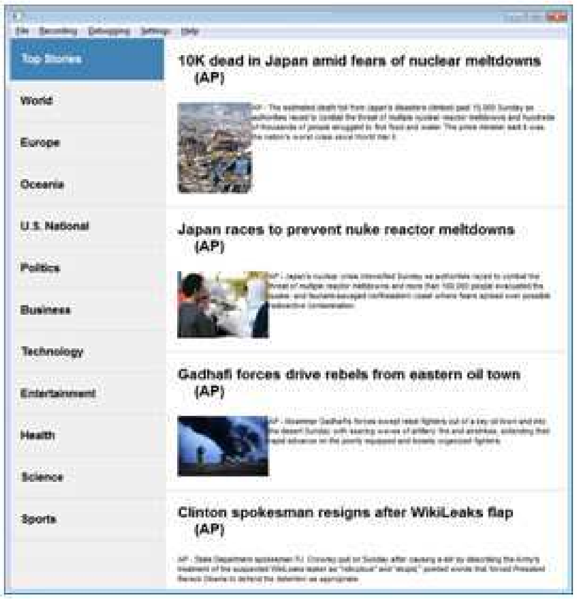
When data downloading is finished, it still has to be processed before displaying the data in the application. One of the possible options is to use QXmlQuery class allowing to convert unformatted XML to the data than can be displayed in QWebView as it is shown below:

```
QXmlQuery query(QXmlQuery::XSLT20);
query.setFocus(&reply)
query.setQuery(&queryFile);
query.evaluateTo(&result);
QWebView *view = new QWebView;
view -> setHtml(QString(result))
```

(Nokia, 2009)

6 QML

Qt WebKit is a powerful engine, which allows to develop high standard hybrid applications. However, following the market demand, as a part of Qt framework, a new language – QML was created, that provides functionality for creating different kind of applications.



10K dead in Japan amid fears of nuclear meltdowns (AP)

[illegible]

Japan races to prevent nuke reactor meltdowns (AP)

[illegible]

Gadhafi forces drive rebels from eastern oil town (AP)

[illegible]

Clinton spokesman resigns after WikiLeaks flap (AP)

[illegible]

FIGURE 19. RSS News Reader QML demo (Nokia, 2011)

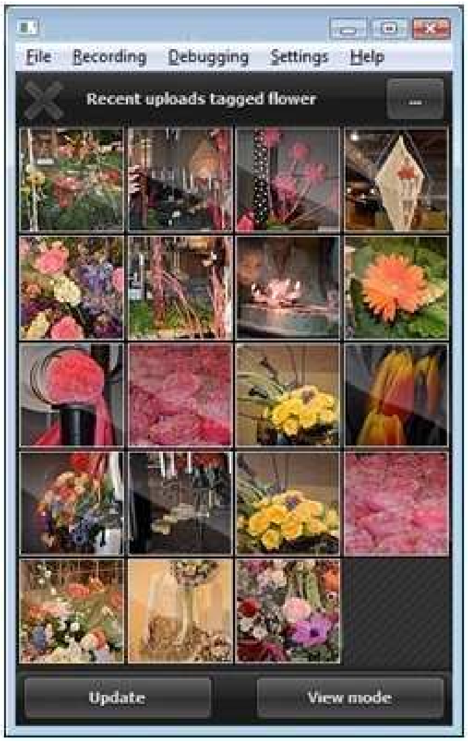


FIGURE 20. Flickr QML demo (Nokia, 2011)

QML is a JavaScript based, declarative language. The latest stable version 4.7.2 was released on the first of March 2011. Its purpose is to design user interfaces mainly for applications where user experience is crucial. QML in the first place focuses on how a UI looks like and behaves. As it is based on JavaScript, it is enough to know it in some degree and maybe some basics of other web technologies, such as HTML and CSS to be able to use QML without any problems. It is also very beneficial, however, to extend QML in C++, which enables, inter alia:

- using functionality defined in C++ source
- accessing functionality in the Qt Declarative module
- writing own QML elements.

(Nokia, 2011)

QML defines the user interface as a tree of objects with properties. The objects are declared with their name and two curly brackets, which contain the object's properties. The QML code example below shows that two objects are created: a `Rectangle` and its child `Text`. The parent- child relationship is a characteristic feature of QML, since it is very common that one object can be nested in the other one. Both of the elements, from the example below, possess a set of properties which defines them. For example an object `Rectangle` has the properties `width` and `height` both with values equal `200`.

```
Rectangle {
    width: 200
    height: 200
    Text {
        x: 66
        y: 93
        text: "Hello World"
    }
}
```

The code fragment above shows only two QML elements, when in practice there are various groups of elements. The basic division of QML elements is presented below:

- **Basic QML Elements** (Item, Component, QtObject)
- **Graphics** (Rectangle, Image, Gradient, etc.)
- **Text Handling** (Text, TextInput, TextEdit, etc.)

- **Mouse and Interaction Area** (MouseArea, Keys, etc.)
- **Positioners and Repeater** (Column, Row, etc.)
- **Transformations** (Scale, Rotation, Translate)
- **States** (State, PropertyChanges, etc.)
- **Animation and Transitions** (Transition, Behavior, SequentialAnimation, etc.)
- **Models and Data Handling** (ListModel, XmlListModel, Package, etc.)
- **Views** (ListView, GridView, PathView)
- **Path Definition** (Path, PathLine, etc.)
- **Utility** (Connections, Qt, etc.)
- **Graphical Effects** (Particles, ParticleMotionLinear, etc.)
- **Add-On Elements** (WebView, Mobility QML Plugin).

(Nokia, 2011)

QML provides also direct access to some concepts from Qt, which are listed below:

- QAction
- QObject signals and slots
- QObject properties
- QWidget
- Qt models (e.g. QAbstractModel).

(Nokia, 2011)

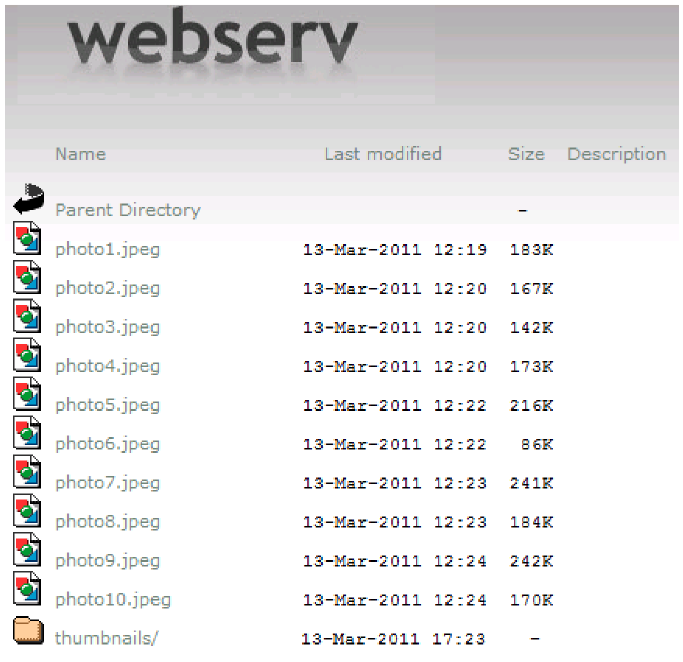
QML is also therefore a good solution for creating hybrid applications. However, despite the fact that it allows to design nice UI and JavaScript takes care of the application's logic, sometimes it still beneficial to use C++. It can extend the application and enrich it with additional functionality, which is very crucial when developing applications that should meet the end user's expectations.

(Nokia, 2011)

7 EXAMPLE OF HYBRID APPLICATION

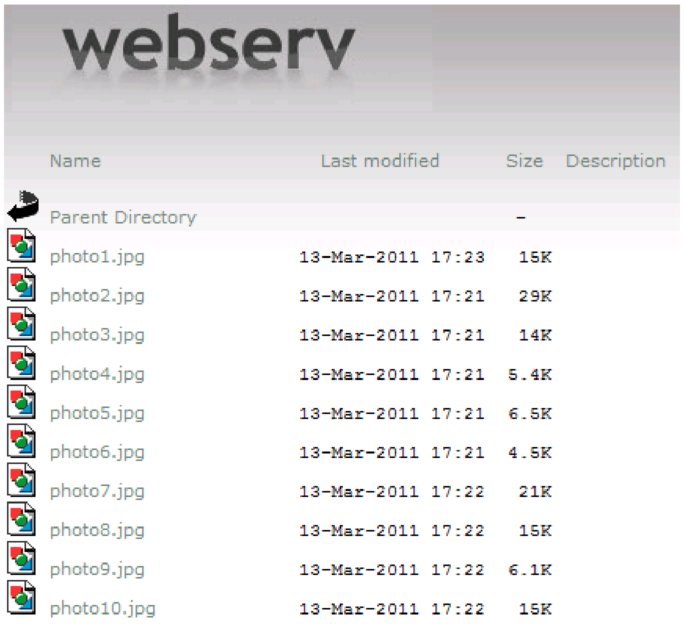
Hybrid applications can be less or more complicated – everything depends on many factors, like: an application's purpose and how far expanded, efficient, fast and secure it is. Below is an example that presents a very simple hybrid application – image gallery.

The main precondition that had to be met before developing the program was setting up a web server. It was created based on the freeware program WebServ (Geeknet, 2003). In the main folder of the web server images and a folder with image's thumbnails were placed, as it is presented in FIGURE 21 and in FIGURE 22.



```
Name                Last modified       Size  Description
Parent Directory                         -
photo1.jpeg         13-Mar-2011 12:19   183K
photo2.jpeg         13-Mar-2011 12:20   167K
photo3.jpeg         13-Mar-2011 12:20   142K
photo4.jpeg         13-Mar-2011 12:20   173K
photo5.jpeg         13-Mar-2011 12:22   216K
photo6.jpeg         13-Mar-2011 12:22    86K
photo7.jpeg         13-Mar-2011 12:23   241K
photo8.jpeg         13-Mar-2011 12:23   184K
photo9.jpeg         13-Mar-2011 12:24   242K
photo10.jpeg        13-Mar-2011 12:24   170K
thumbnails/         13-Mar-2011 17:23     -
```

FIGURE 21. Web server main folder


webserv

Name	Last modified	Size	Description
Parent Directory		-	
photo1.jpg	13-Mar-2011 17:23	15K	
photo2.jpg	13-Mar-2011 17:21	29K	
photo3.jpg	13-Mar-2011 17:21	14K	
photo4.jpg	13-Mar-2011 17:21	5.4K	
photo5.jpg	13-Mar-2011 17:21	6.5K	
photo6.jpg	13-Mar-2011 17:21	4.5K	
photo7.jpg	13-Mar-2011 17:22	21K	
photo8.jpg	13-Mar-2011 17:22	15K	
photo9.jpg	13-Mar-2011 17:22	6.1K	
photo10.jpg	13-Mar-2011 17:22	15K	

FIGURE 22. Web Server thumbnails folder

The main purpose of the application is to show how web content is retrieved from the server and displayed in a native application. It is done mainly using QWebView and JavaScript. JavaScript is responsible for basic functionality, such as displaying images and thumbnails, transition and user interaction with thumbnails. Native core is responsible for creating QWebView and loading a page from demo.htm.

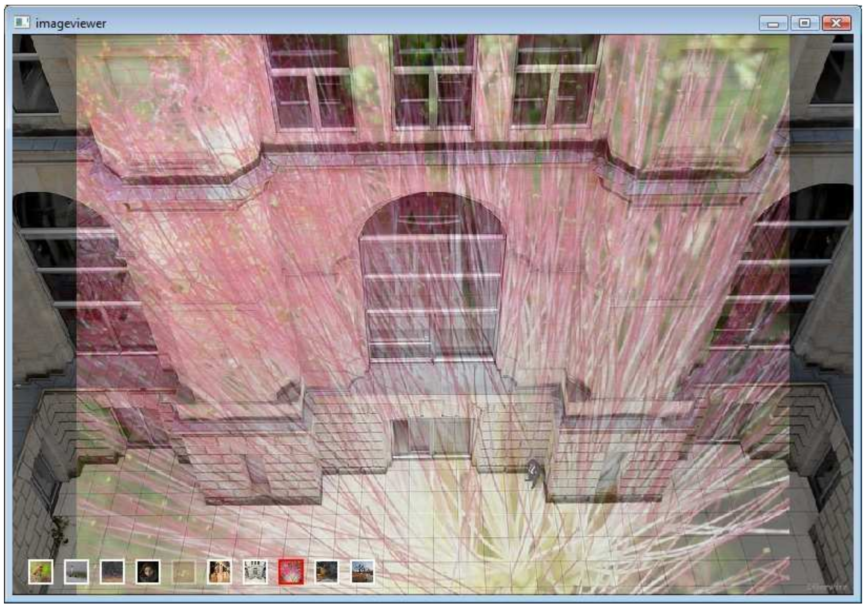


FIGURE 23. Transition between two images in the imageviewer application run on Microsoft Vista

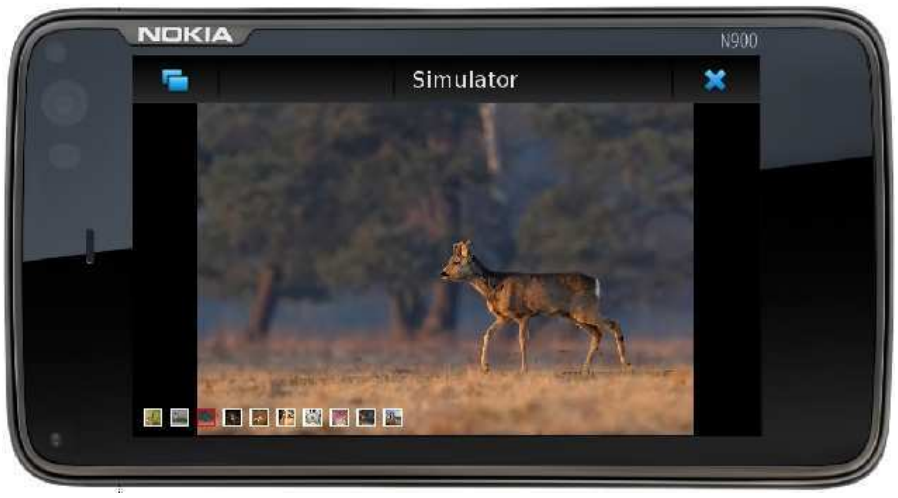


FIGURE 24. ImageViewer application run in the Qt Simulator

8 CONCLUSION

The main goal of this Bachelor’s thesis was the explication of the hybrid application concept and basic aspects helping in its optimal development. I wanted to present both the advantages and problems that each developer needs to face while creating such as application. All presented information aims at highlighting how powerful hybrid applications can be but only when they are used in a proper way and for creating an application that uses all benefits of hybridization.

Nowadays Internet is one of the most powerful and the biggest source providing a very large amount of both information and services. Among the offered services there are the biggest and the most popular ones, like: Facebook, Flickr, Twitter, MySpace, Google Maps, Picasa, Ebay, Amazon etc. All of these kinds of services provide a huge amount of content that is not so easily accessible from mobile devices, mainly due to lack of a mouse, a keyboard and large enough display, which for example a common computer has. Limitations of mobile devices are very often a reason for which an average user has problems with using such services through mobile phones’ web browsers. This is, however, the opportunity for hybrid applications which allow creating an application containing native functionality that is used for building user-friendly interface and providing an easier access to a chosen service. Therefore a very popular, common service can be still used on mobile devices but in a much easier way than when it would have to be accessed through a browser.

The other benefit from creating hybrid applications is that a major part of the application is stored on the web servers. This approach brings many advantages, like lighter application and hence faster installation and also less complex updates.

The next thing worth mentioning is the possibility of reduction of demand for computing power. Applications installed on the device can contain most of the functionality, however, all very complex calculations can be executed on the powerful, fast servers and only the final result can be resent to the actual application.

On the other hand, developing hybrid applications can entail various problems. The largest problem is a network connection lost, which causes that the applications' functionality cannot be fully used. The applications, however, can show for example the last active state, or a part of the functionality can be used. For example if Twitter is used as web content, the solution can be that the user can still write some posts which are kept on hold and they are sent only when the connection is restored.

There are many other problems that have to be taken into account when designing a hybrid application, like security of the application, which is limited due to the web content used in the application or speed of the internet connection. These issues, however, can be solved or bypassed and they depend a lot on the type of the application's content. What is worth considering is the reliability of the web services' providers. Hybrid applications rely on web services, which in the extreme situations can be malfunctioned, down, hacked or even shut down and then the application related to this service becomes useless.

The examples above show that hybrid applications can be the future of mobile devices, because they can bypass the limitations of such devices by putting most of the functionality on the web servers. On the other hand there are still many problems that have to be considered when designing and developing hybrid applications. Pure native or web applications are still very good solutions in many situations, that is why the key for using all benefits of hybridization is to take into account all aspects of using and maintaining such applications and then they can became a strong competitor amongst other applications on mobile devices' market.

9 REFERENCES

Chaffee, A. (2000, August 18). *jGuru*. Referenced March 14, 2011, from What is a web application (or "webapp")?: http://www.jguru.com/faq/view.jsp?EID=129328

Dhandhania, A. (2011). *HTML 5: Client-side Storage*. Referenced March 9, 2011, from http://www.webreference.com/authoring/languages/html/HTML5-Client-Side/

Digia Plc. (2011). *digia*. Referenced March 12, 2011, from http://www.digia.com/

dotMobi. (2010). *mobiThinking*. Referenced March 12, 2011, from Mobile applications: native v Web apps – what are the pros and cons?: http://mobithinking.com/native-or-web-app

Geeknet. (2003). *WebServ*. Referenced March 14, 2011, from http://webserv.sourceforge.net/

Johnson, D. (2008). *ICS Network.* Referenced March 11, 2011, from Wt WebKit: http://www.ics.com/learning/icsnetwork_flash/cd4497f03840f43841a5e5917bb2ef26

Kosonen, P. (2009). *Next Generation Hybrid Applications with Qt.* Referenced March 2, 2011, from http://www.slideshare.net/pkosonen/next-generation-hybrid-applications-with-qt-presentation-for-see-2009

Lal, R. (2009). Referenced March 5, 2011, from Hybrid Application Development for Maemo N900 Device using Qt Webkit: http://www.slideshare.net/rajeshlal/hybrid-application-development-for-maemo-n900-device

Mills, E. (2009). *AVAI Mobile Solutions Blog*. Referenced March 12, 2011, from iPhone App Decisions - Native App, Web App, or Hybrid App: http://avaimobile.com/blog/bid/17266/iPhone-App-Decisions-Native-App-Web-App-or-Hybrid-App

Nokia. (2009). Referenced March 11, 2011, from Qt Features for Hybrid Web / Native Application Development: http://qt.nokia.com/qt-in-use/files/pdf/qt-features-for-hybrid-web-native-application-development

Nokia. (2009). Referenced March 7, 2011, from Developments in Qt WebKit Integration:

http://get.qt.nokia.com/videos/DevDays2009/Technical%20Sessions/DevDays2009%20-%20Developments%20in%20the%20Qt%20WebKit%20Integration.pdf

Nokia. (n.d.). *Applications using QtWebKit*. Referenced March 12, 2011, from http://developer.qt.nokia.com/wiki/Applications_Using_QtWebKit

Nokia. (2010). *Build Amazing Mobile Apps using HTML5, CSS3 and JavaScript.* Referenced March 1, 2011, from http://conference2010.meego.com/sites/all/files/sessions/amazingmobileappswithhtml5css3javascript.pdf

Nokia. (2011). *Qt*. Referenced March 12, 2011, from http://qt.nokia.com/

Nokia. (2011). *Qt 4.7: Using QML in C++*. Referenced March 20, 2011, from Qt 4.7: http://doc.qt.nokia.com/4.7-snapshot/qtbinding.html

Nokia. (2011). *Qt Creator Whitepaper*. Referenced March 12, 2011, from http://developer.qt.nokia.com/wiki/QtCreatorWhitepaper#5ec21dff0f0166c5a8c09d19630488d9

Nokia. (2011). *Qt Reference Documentation*. Referenced March 12, 2011, from QML Examples and Demos: http://doc.qt.nokia.com/4.7-snapshot/qdeclarativeexamples.html

Ovisoftware. (2010). *Ovi Software*. Pobrano March 12, 2011 z lokalizacji Qt Development - MeeGo: http://www.ovisoftware.com/meego-qt-development.html

Panayara. (2011, March 1). *Linux and Microcontroller Tips*. Referenced March 14, 2011, from Qt: http://shibuvarkala.blogspot.com/2011/03/download-free-qt-training-course.html

Pavley, J. (2010). *TechNewsWorld*. Referenced March 12, 2011, from Hybrid Apps: The Art of Being in Two Places at Once: http://www.technewsworld.com/story/70716.html

Sharp, R. (2010). *<html>5doctor*. Referenced March 11, 2011, from Introducing Web SQL Databases: http://html5doctor.com/introducing-web-sql-databases/

The WebKit Open Source Project. (2011). Referenced March 3, 2011, from http://www.webkit.org/

Watt, A., Watt, J., Simon, J., & ODonnell, J. (2003). *JavaScript dla każdego.* Helion.