Gather data from web page with JavaScript, WebKit, and Qt

Overview

This snippet demonstrates how you can gather data from a web page to Qt via WebKit. We will use the Display local web page with Qt WebKit article as base for this snippet.

Preconditions

- Read Display local web page with Qt WebKit

Code

html/view.html

We'll add two JavaScript includes in the head tag.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <!-- You reference the resources with qrc:// -->
    <!-- Stylesheet reference -->
    <link rel="stylesheet" href="qrc:///css/view.css" />
    <!-- jQuery include -->
    <script type="text/javascript" src="qrc:///js/jquery.js" />
    <!-- Our own script include -->
    <script type="text/javascript" src="qrc:///js/script.js" />

    <title>WebKit example</title>
  </head>
  <body>
    <!-- Header -->
    <h1 id="header">Welcome</h1>
    <!-- Paragraph -->
    <p id="paragraph">Hello World!</p>
    <!-- Form -->
    <form id="form" action="#">
      <fieldset>
        <label for="text">Text</label>
        <!-- Text input -->
        <input id="text" type="text" />
        <!-- Submit button -->
        <input id="button" type="button" value="Submit!"/>
      </fieldset>
    </form>
  </body>
</html>
```

js/jquery.js

This js file is a reference to a JavaScript library called jQuery. You can obtain it here: http://docs.jquery.com/Downloading_jQuery#Download_jQuery

jQuery enables you to be more productive when writing JavaScript.

js/script.js

This file includes JavaScript which is used to create animations and dynamically add content to the site from the web page.

```
/**
  * Adds p element after the element which has paragraph as the id
  * with the content from input element with text as the id.
  * This is jQuery syntax: http://www.jquery.com
  *
  * $("#paragraph") documented here: http://docs.jquery.com/Selectors/id#id
  * .after documented here: http://docs.jquery.com/Manipulation/after
  * .val documented here: http://docs.jquery.com/Val
  */
button_click = function(event) {
    $("#paragraph").after("<p>" + $("#text").val() + "</p>");
}

/**
  * This is run after the the web page has been rendered.
  * $(document).ready documented here:
  * http://docs.jquery.com/Tutorials:Introducing_$(document).ready()
  */
$(document).ready(function(){
    /*
     * Hides all the h1 elements.
     * Element selector: http://docs.jquery.com/Selectors/element#element
     * hide function: http://docs.jquery.com/Hide
```


(C) Copyright Nokia 2013. All rights reserved.

```
     */
    $("h1").hide();
    /*
     * Hides all the p elements.
     */
    $("p").hide();
    /*
     * Hides element with form id.
     */
    $("#form").hide();
    /*
     * Shows all the h1 elements with animation.
     * show function: http://docs.jquery.com/Show
     */
    $("h1").show('slow');
    /*
     * Shows all the p elements with animation.
     */
    $("p").show('slow');
    /*
     * Shows the element with form as id with animation.
     */
    $("#form").show('slow');
    /*
     * Adds button_click as click event handler
     * for element with button as id.
     * click function: http://docs.jquery.com/Events/click
     */
    $("#button").click(button_click);
});
```

src/gather-data.js

```
/**
  * Gathers all the text content from p elements
  * to an array and returns it.
  */
Qt_QWET_gather = function() {
  a = [];
  e = document.getElementsByTagName('p');
  for(var i = 0; i < e.length; i++) {
    a.push(e[i].textContent);
  }
  return a;
};
Qt_QWET_gather();
```

Add these files to the resources file and include the resource file in your project file as explained in Using resources in Qt.

The following code demonstrates how to read data from the web page via JavaScript executed from Qt.

src/qwet.h

The only things added to the header file in comparison to Display local web page with Qt WebKit are includes, the `gatherButtonClicked` slot, helper methods, and a reference to `QListWidget` which is used to display the gathered data.

```
#ifndef QWET_H
#define QWET_H

#include <QtCore/QPointer>
#include <QtCore/QFile>
#include <QtCore/QTextStream>
#include <QtCore/QVariant>
#include <QtCore/QList>

#include <QtWebKit/QWebView>
#include <QtWebKit/QWebPage>
#include <QtWebKit/QWebFrame>

#include <QtGui/QWidget>
#include <QtGui/QMainWindow>
#include <QtGui/QFrame>
#include <QtGui/QVBoxLayout>
#include <QtGui/QListWidget>
#include <QtGui/QPushButton>

class QWET : public QMainWindow
{
    Q_OBJECT

public:
    QWET(QWidget *parent = 0);
    ~QWET();

private slots:
    void gatherButtonClicked();
private:
    QPointer<QWebView> _webView;
    QPointer<QVBoxLayout> _layout;
    QPointer<QListWidget> _pList;

    QString _gatherJS;

    void setupUI();

    QString readFileToQString(const QString& fileName) const;
    void updatePList(QListWidget* widget,
                     const QList<QVariant>& values);
};

#endif // QWET_H
```

src/qwet.cpp

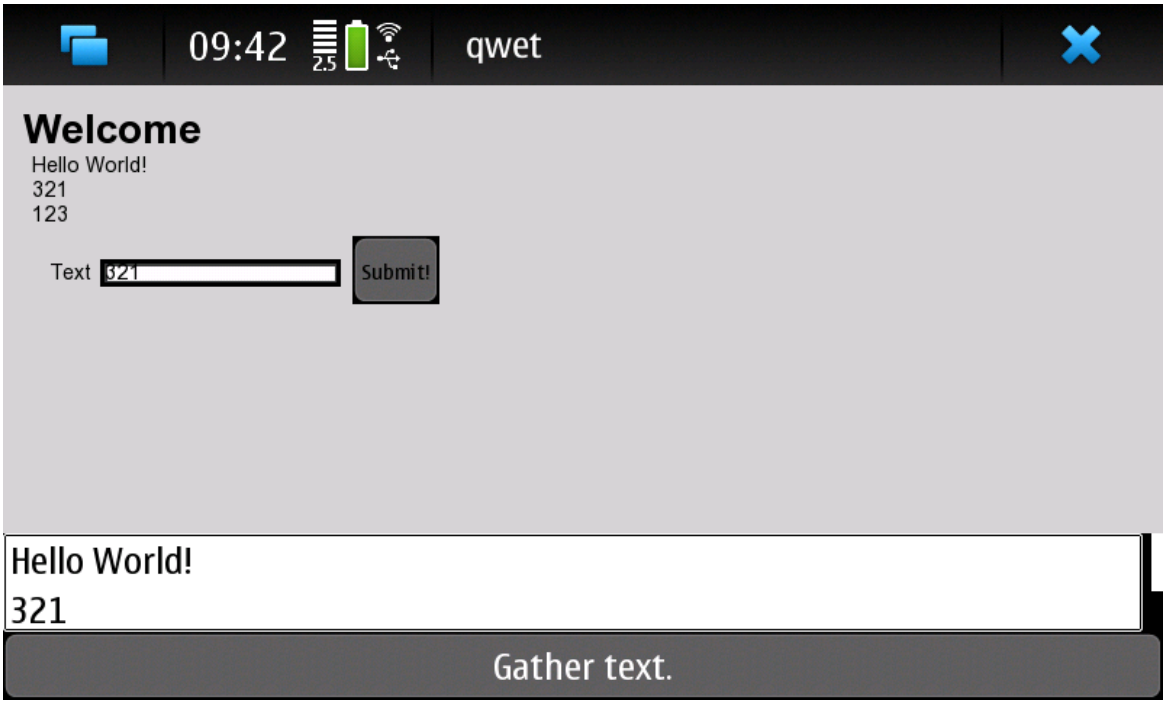
(C) Copyright Nokia 2013. All rights reserved.

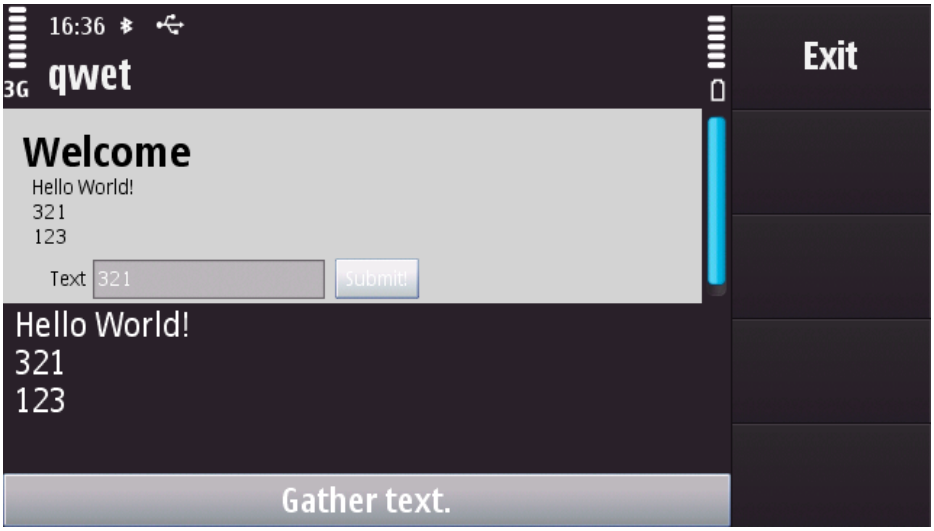
```
#include "qwet.h"

QWET::QWET(QWidget *parent) : QMainWindow(parent)
{
    setupUI();
}

QWET::~QWET()
{
}

void QWET::setupUI()
{
    /**
      * Create the central widget
      * and set it.
      */
    QFrame* cW = new QFrame(this);
    setCentralWidget(cW);

    /**
      * Set the layout to central widget.
      */
    _layout = new QVBoxLayout(this);
    cW->setLayout(_layout);
    _layout->setMargin(0);
    _layout->setSpacing(0);

    /**
      * Let's create the web view which
      * will be used to display our page
      * with which we will be communicating with.
      */
    _webView = new QWebView(this);
    _webView->load(QUrl("qrc:///html/view.html"));
    _webView->show();

    /** Add it to layout */
    _layout->addWidget(_webView);

    /** We'll gather the results to list widget.  */
    _pList = new QListWidget(this);
    _layout->addWidget(_pList);

    /** Add the gather text button */
    QPushButton* gatherButton = new QPushButton("Gather text.", this);
    connect(gatherButton, SIGNAL(clicked()),
            this, SLOT(gatherButtonClicked()));

    _layout->addWidget(gatherButton);
}

/**
  * This slot gathers the p text content from the html page
  * and adds it to the the list widget.
  */
void QWET::gatherButtonClicked()
{
    /**
      * Read the java script to be executed to a string.
      */
    if(this->_gatherJS.isEmpty()) {
        this->_gatherJS = this->readFileToQString(QString(":/js/gather-data.js"));
    }
    else {
        this->_gatherJS = "Qt_QWET_gather();";
    }
    /** Run the java script on the page and get results. */
    QVariant variant = this->_webView->page()->mainFrame()->evaluateJavaScript(this->_gatherJS);
    /** We make sure that the results are there. */
    if(!variant.isNull() &&
       variant.type() == QVariant::List) {
        /** Ask for the acual list we're expecting. */
        QList<QVariant> list = variant.toList();
        /** Update the entries in the list widget.*/
        this->updatePList(this->_pList.data(), list);
    }
}

/**
  * This function updates list widgets entries
  */
void QWET::updatePList(QListWidget* widget, const QList<QVariant>& values)
{
    /** Check that the parameters are valid. */
    if(widget != 0 &&
       !values.isEmpty()) {
        /** Clear the entries. */
        widget->clear();
        /** Go through the values...*/
        QListIterator<QVariant> i(values);
        while(i.hasNext()) {
            QVariant v = i.next();
            /** ...and add them to widget. */
            widget->addItem(v.toString());
        }
    }
}

/**
  * Reads file to a string.
  */
QString QWET::readFileToQString(const QString& fileName) const
{
    QFile file(fileName);
    if (!file.open(QIODevice::ReadOnly | QIODevice::Text))
        return QString();

    QTextStream in(&file);
    QString content = in.readAll();
    file.close();
    return content;
}
```

Postconditions

(C) Copyright Nokia 2013. All rights reserved.

You've now created this:





See also

- Display local web page with Qt WebKit
- Add data to a web page with JavaScript, WebKit, and Qt

(C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.