Browser Control API

This article provides and overview of the Symbian Browser Control API, which can be used to display web content from within an application.

Introduction

The browser for Symbian enables mobile phone users to browse the World Wide Web. Web pages can be implemented in Hypertext Markup Language (HTML), Extended Hypertext Markup Language (XHTML), or Wireless Markup Language (WML).

A **Browser Control** is a control that you embed into an application to enable users to view Web content from within that application. The host application can control the way the **Browser Control** displays the Web content, dialogs, softkeys, scroll bars, and so on.

The **Browser Control API** enables developers to do the following:

- Display XHTML, HTML, WML, CSS, or scripted content inside an application
- Customize the Browser Control for your specific application. For example,
  - Draw scroll bars
  - Design options menus
  - Display notification messages
  - Customize softkeys
  - Change the display size of a Web page

Environmental Requirements

The **Browser Control API** is compatible with any application that complies with S60 3rd Edition.

Standards Compliance

The **Browser Control API** complies with the following standards:

- ECMA Script Mobile Profile
- HTML 4.0.1
- Java Script
- Wireless CSS
- WML 1.3
- WML Script
- XHTML Mobile Profile

Browser Control Functionality

Basic Browser Control functionality includes:

- Downloading and displaying Web content
- Display application content, through (***LoadDataL*** function)
- Resolving links
- Retrieving the following information about the downloaded content:
- Display size needed to see all of the content (***ContentSize*** function)
- WML options associated with the current page(***WMLOptionMenuItemsL*** function)
- Value of a specified browser setting (***BrowserSettingL*** function)
- Indication of whether or not a particular URL is in the cache (***IsUrlInCache*** function)
- Information about the page, such as its title, URL, and content (***PageInfoLC*** function)
- Indication of whether there is a previous or next element in the history stack (***NavigationAvailable*** function)
- Build version of the Browser Control (***VersionInfoLC*** function)
- Number of images in the current page (***ImageCountL*** function)
- Type of focused element (***FocusedElementType*** function)
- Minimum size of the Browser Control (***MinimumSize*** function)

Special Features

The Browser Control API supports the use of URI with fragment identifier. Fragment identifier is a string that goes in the end of a URI and identifies something specific to the document addressed by the URI. The Fragment Identifier and the rest of the URI are separated by a '#' symbol (Ex.: http://foo.bar/index.html#fragment_id or http://foo.bar/fragment_id). For more information see Fragment Identifier

The Browser Control API supports the use of fragment identifier. An example of use is a html file downloaded in phone memory (c:/Data/test.html).

```
<html>
<head>
 <!-- details omitted -->
</head>

<body>

<div id="fragment01">Fragment 01</div>

<div id="fragment02">Fragment 02</div>

<div id="fragment03">Fragment 03.</div>

</body>
</html>
```


(C) Copyright Nokia 2013. All rights reserved.

To load such a file using the Browser Control API one should use the following code.

```
void CTestBrowserAPIAppView::ConstructL( const TRect& aRect )
{
    // Create a window for this application view
    CreateWindowL();

    // Set the windows size
    SetRect( aRect );


    TRect rect( aRect.iTl, TSize( Size().iWidth, 20 ));
    iBrCtlInterface = CreateBrowserControlL( this, rect, TBrCtlDefs::ECapabilityDisplayScrollBar |
                                             TBrCtlDefs::ECapabilityLoadHttpFw,TBrCtlDefs::ECommandIdBase );
    iBrCtlInterface->LoadUrlL(_L("file://c:/Data/test.html#fragment02"));

    // Activate the window, which makes it ready to be drawn
    ActivateL();
}
```

Getting Started with the Browser Control

When using the browser control in your application, you need to use the `CBrCtlInterface` interface. For this interface you need to include brctlinterface.h in your cpp file.

Include

```
#include <brctlinterface.h>
```

you also need to link against browserengine.lib

which you can do by putting the following in your MMP file.

Library

```
LIBRARY browserengine.lib
```

The file brctrlinterface.h is filled with usefull comments, its probably better to read that file in order to get more familiar with the browser control creation and usage.

To create an instance of the browser control you need to use the `CreateBrowserControlL()` function. This function takes quite a lot of parameters, however all of these are explained in the Browser Control API Developers Guide.

The third parameter to this function is `TBrCtlDefs::TBrCtlCapabilities`, this is an enum which is defined in the brctldefs.h file. Where `TBrCtlDefs` is a C++ class that holds all the browser defines etc.

Related Links

For more information see: Browser Control API Developer's Guide

Fragment Idenfifier reference from w3

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.