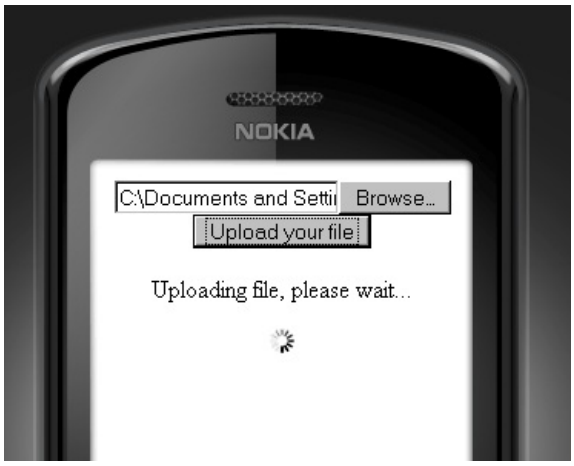
How to upload files from Symbian Web Runtime

This article explains how to upload files from a Web Runtime widget.



Description

File uploading functionality is often useful in mobile applications to allow users to upload files from their mobile devices to a remote server. A detailed overview of File Upload is available on this Nokia Developer Wiki article: Mobile Design Pattern: File Upload.

Benefits

The ability to upload files from within a WRT widget allows to build more useful and richer widgets for the final users. The proposed approach has the following benefits for the user experience of a WRT widget:

- file upload is managed from the widget itself, without the need for the user to perform extra steps
- file upload is performed in a background, asynchronous operation. So, while the file is uploading, the user can continue to use the widget without having to wait for its completion
- multiple uploads can be handled, so allowing to upload more files at once, without repeating all the required steps for each file
- with appropriate client and server-side code, the user can know the exact status of each upload (so, the percentage of file already uploaded, and how much it takes to finish)

How to

Current Web Runtime APIs do not offer a direct way to upload files through JavaScript. So, only available option is to use an approach widely used on the Web, in order to implement asynchronous file uploading, without letting the user leave the current page.

IFrames

This is accomplished by using the <iframe> tag, that allows to embed separate windows inside the main widget's window.

For more information about IFrame, its usage and limits when used within Web Runtime widgets, these resources are available:

- IFrames on Nokia Developer Library
- S60 Widgets and iframes Wiki article

File upload implementation

The HTML code

The HTML code needed in order to implement file upload consists of 3 main elements:

1. the HTML form containing an input field of type file, to allow the user to choose a file from his device
2. an hidden IFrame, that will be used as target for the HTML form
3. a loading indicator, that will be shown during the upload process

So, the it's possible to define the following code:

(C) Copyright Nokia 2014. All rights reserved.

```
<html>

 [...]

 <body>

  <div id="page">
   <form onsubmit="return startUpload();" target="upload_target" method="post"
enctype="multipart/form-data" action="http://www.jappit.com/uploadtest.php" >
    <input type="file" name="file_field" id="file_field"/>
    <br/>
    <input type="submit" value="Upload your file" />
   </form>

   <div id="upload_indicator" style="display:none">
    <p>Uploading file, please wait...</p>
    <img src="images/ajax-loader.gif" />
   </div>
  </div>

  <iframe name="upload_target" id="upload_target" style="display: none;"></iframe>
 </body>
</html>
```

Few things to note in the above code:

- The target attribute of the HTML form is set equal to the iframe name
- In order to correctly upload the file, the enctype="multipart/form-data" is specified in the <form> tag
- The form onsubmit event will call the `startUpload()` method, that will be defined below
- The loading indicator (`upload_indicator`) is initially hidden, and will be shown when the upload starts
- The IFrame element is also hidden, since it must not be visible in the widget's user interface

The server side code

Before digging in the widget's JavaScript code, a server side script must be defined in order to handle the uploaded file and to send to the WRT widget the upload result.

The following PHP script will accept files smaller than 10 Kb (returning 1 to the widget), rejecting bigger files (in this case, the 0 (zero) value is returned).

```
<?

<?

if(filesize($_FILES['file_field']['tmp_name']) < 1024 * 10)
{
 //let's save the uploaded file
 move_uploaded_file($_FILES['file_field']['tmp_name'], 'uploadtest.file');

 //and now just delete it
 unlink('uploadtest.file');

 echo 1;
}
else
{
 echo 0;
}
```


(C) Copyright Nokia 2014. All rights reserved.

```
?>


?>
```

The JavaScript code

When the form is submitted, by pressing its submit button, the `startUpload()` function is called. This function should do these things:

1. set an appropriate onload event for the hidden iframe
2. show the loading indicator
3. return the "true" value, in order for the form to be submitted

Also, the `startUpload()` method will check if the file field is empty, and will prompt the user to choose a file.

```
function startUpload()
{
 var fileField = document.getElementById('file_field');

 if(fileField.value.length > 0)
 {
  var targetFrame = document.getElementById('upload_target');

  targetFrame.onload = uploadHandler;

  var uploadIndicator = document.getElementById('upload_indicator');

  uploadIndicator.style.display = '';

  return true;
 }
 else
 {
  alert("Please specify a file to be uploaded");

  return false;
 }
}
```

So, what happens is that, once the form is submitted, the widget starts to upload the file to the remote server, while showing the loading indicator to the user.

When the upload finishes, the iframe onload handler is called. What this function has to do is:

1. hide the loading indicator
2. check the response of the server side script, by reading the IFrame content
3. notify the user about the result of the file upload

```
function uploadHandler()
{
 var uploadIndicator = document.getElementById('upload_indicator');

 uploadIndicator.style.display = 'none';

 var targetFrame = document.getElementById('upload_target');
```


(C) Copyright Nokia 2014. All rights reserved.

```
  var result = document.getElementById('upload_target').contentDocument.body.innerHTML;

  if(result == 1)
  {
   alert("Upload complete!");
  }
  else
  {
   alert("Error: file too big!");
  }

 }
```

Considerations

Improvements to the code presented in this article can be done in order to:

- allow multiple file uploads (check this Nokia Developer Wiki article for more information about multiple file uploads: Mobile Design Pattern: File Upload)
- give to the user information about completed upload percentage, further improving the user experience for the final user, since he will exactly know how much of his file is already uploaded, and how much it could take to complete it. Since the client has no direct access to this information, in order to accomplish this, both the server and the widget code must be modified:
  - the widget must poll the server at predefined intervals, to retrieve the current uploaded percentage
  - the server must reply to the client returning the requested value

Downloads

The script developed in this article is available for download here: Media:FileUploadWidget.zip

(C) Copyright Nokia 2014. All rights reserved.