Mobile Design Pattern: File Upload

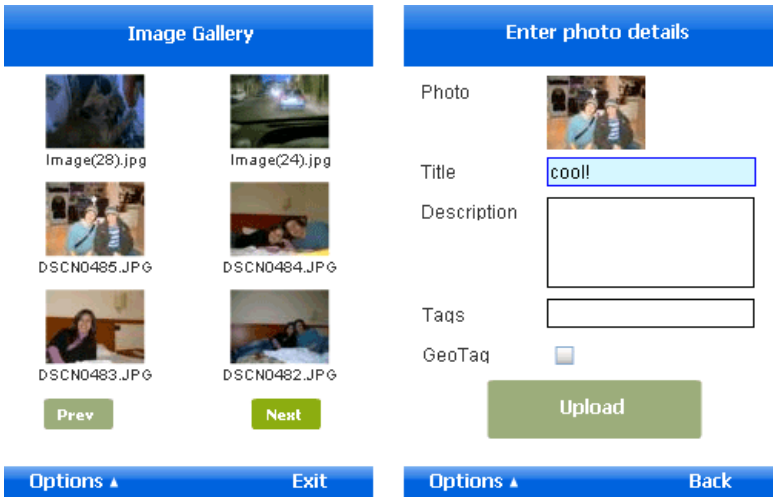
Description

File uploads allow users to **upload files from their devices to a remote server**, directly from a mobile application.

Single file upload

The simplest scenario is the **upload of a single file**. The user is typically required to **select a file**, by browsing files available on his device, and then **confirm the upload** by pressing a button or softkey. Sometimes, the user is allowed to compile some other **meta-data fields**, with data that will be sent to the remote server together with the uploaded file.

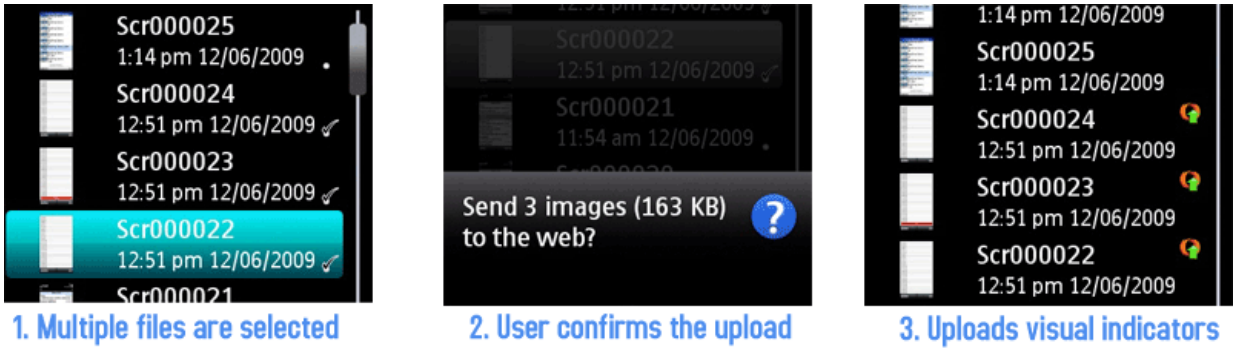
An example of single file upload is visible in the Flikun client, a photo uploader for Flickr:



Multiple files upload

A more complex approach to file uploads is represented by the **upload of more files**. The user is requested to choose one or more files from his device, and then starts the upload. Typically, multiple files upload is **supported by detailed progress information about the upload of each file**.

An example of multiple file uploads is visible in the **Shozu mobile client**:



1. Multiple files are selected
2. User confirms the upload
3. Uploads visual indicators

Upload progress status

Both in single and multiple file uploads, it is **good practice** to give to the user a **clear indication of the upload progress status**.

(C) Copyright Nokia 2014. All rights reserved.

This can be done in different ways, depending on the available information about the upload status:

- If the **percentage of the file already uploaded** is known, a visual representation of this percentage is shown (e.g.: by using a progress bar or a similar indicator).
- If there's no information about the uploaded percentage, then a **generic animated indicator** is used, to notify the user about the ongoing upload

For **multiple file uploads**, a **dedicated view** is typically used to give detailed information about the status of each upload. This view should give to the user clear information about **which files have already been uploaded**, which are **currently uploading**, and which are **waiting to be uploaded**.

Detailed explanation of **progress and wait indicators** is available on this Nokia Developer Wiki article: Mobile Design Pattern: Progress and Wait Indicator

Use when

File uploads are useful in all those situations where a file must be transferred to a remote server. Sample scenarios include:

- a **mobile client for a community**, that offers to mobile users to upload photos and videos directly from their devices
- a client allowing users to **backup files on a remote server**
- **FTP clients**

Use how

File uploads can be **differently handled and implemented depending on the technology** used in a mobile application. Since file uploads require the capability for the users to choose one or more files from their devices, the used technology must provide some sort of **access to the underlying file system**.

An usual approach to implement file uploads is represented by **HTTP MultiPart** **POST requests**, the same method used by website forms to upload files. Other possible approaches include **FTP transfers**, or direct **Socket connections**.

Java ME

Java ME applications can allow users to upload files by:

- using **JSR 75 FileConnection API** for file browsing
- using **HttpConnection** or **SocketConnection** instances to transfer the file bytes to the remote server

An example of file uploads in Java ME that uses the HttpConnection object is available in this Nokia Developer Wiki article: HTTP Post multipart file upload in Java ME

Flash Lite

Flash Lite **does not natively support functionalities that allow file uploads**. But this is possible, on specific subsets of devices, through **third-party plugins**.

An example is represented by KuneriLite, that allows file uploads by using the following plugins:

- The File Plugin to allow the user to **browse and choose a file** from his device
- The Upload/Download plugin to **upload files** to the remote server

An example of file upload implemented with Flash Lite and the KuneriLite plugins is the Flikun client.

Symbian C++

A detailed example showing how to upload files in Symbian C++, by using the multipart approach, is available here: Symbian C++ : Multipart/form-data

Python

The following articles show how to **browse files** on device's file system, and how to **upload them to a remote server** in Python:

- File Selection in PySymbian
- How to upload a file to server with application/x-www-form-urlencoded
- How to upload a file to server with multipart/form-data

Web Runtime

(C) Copyright Nokia 2014. All rights reserved.

The following article explains how to **upload files** from a **Web Runtime widget**: How to upload files from a Web Runtime widget

Design tips

- While performing a file upload, leave the **user interface responsive**, to allow the **user to continue using the application**
- Give the user the possibility to **cancel an ongoing upload** with a clearly visible cancel option
- Notify the user about the **upload progress status**

(C) Copyright Nokia 2014. All rights reserved.