

# Mobile Web Server: Mashup Application Design

Version 1.0; May 29, 2007

# Mobile Web Server

**NOKIA**

Copyright © 2007 Nokia Corporation. All rights reserved.

Nokia and Forum Nokia are registered trademarks of Nokia Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

#### **Disclaimer**

The information in this document is provided “as is,” with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. This document is provided for informational purposes only.

Nokia Corporation disclaims all liability, including liability for infringement of any proprietary rights, relating to implementation of information presented in this document. Nokia Corporation does not warrant or represent that such use will not infringe such rights.

Nokia Corporation retains the right to make changes to this specification at any time, without notice.

#### **License**

A license is hereby granted to download and print a copy of this specification for personal use only. No other license to any other intellectual property rights is granted herein.

## Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b> .....                | <b>5</b>  |
| 1.1      | About this document.....                 | 5         |
| 1.2      | Target audience.....                     | 5         |
| <b>2</b> | <b>Mashup overview</b> .....             | <b>6</b>  |
| <b>3</b> | <b>Application overview</b> .....        | <b>6</b>  |
| 3.1      | Usage.....                               | 6         |
| 3.1.1    | Internet API.....                        | 6         |
| 3.1.2    | Application.....                         | 7         |
| 3.1.3    | Connections.....                         | 9         |
| 3.2      | Main functions.....                      | 10        |
| 3.2.1    | Store URLs.....                          | 10        |
| 3.2.2    | Display images from remote location..... | 10        |
| 3.2.3    | Find images.....                         | 11        |
| 3.3      | Class descriptions.....                  | 11        |
| 3.4      | Method descriptions.....                 | 11        |
| 3.4.1    | handler(req).....                        | 11        |
| 3.4.2    | handler(self, req).....                  | 11        |
| 3.4.3    | __init__(self).....                      | 11        |
| 3.4.4    | parse_args(self, req).....               | 11        |
| 3.4.5    | Walk(self, top).....                     | 11        |
| <b>4</b> | <b>Terms and abbreviations</b> .....     | <b>12</b> |
| <b>5</b> | <b>Evaluate this resource</b> .....      | <b>13</b> |

## Change history

|              |             |                          |
|--------------|-------------|--------------------------|
| May 29, 2007 | Version 1.0 | Initial document release |
|              |             |                          |

# 1 Introduction

## 1.1 About this document

This document describes the mashup example application for Mobile Web Server (MWS). The mashup example application is a two-part Web application that demonstrates the usage of multiple data sources when generating mobile Web sites. The two parts are:

- The first part is the content provider server's Internet API. This could be any third-party Internet API but in this example application an example Internet API is used. This Internet API is a Web application in its own but it returns values that can only be comprehended when the second part interprets them. The Internet API has in this example only one function that returns the file names of images that it finds from a specified folder on the content provider server.
- The second part of the mashup example application is the content consumer server Web application that generates a Web site from the data that the content provider server returns. In the example the generated site contains the images found from the content provider server. The Web user can't tell where the images come from without reading the page source code. This technique could be used to create an information hub that combines data from multiple sources into one Web site. Figure 1 illustrates how the mashup demo works.

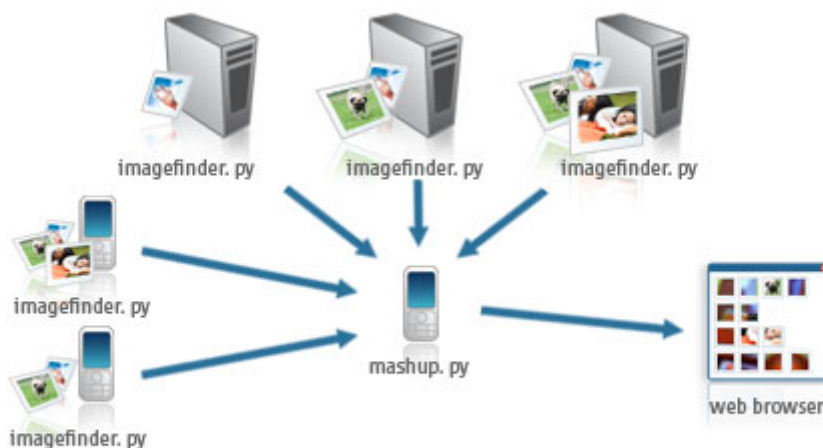


Figure 1: Model of the mashup demo

## 1.2 Target audience

This document is meant for content developers who wish to create mashup applications for the Mobile Web Server.

## 2 Mashup overview

The mashup application is a two-part system that uses multiple sources to generate Web sites. The Web application seen by the Web user calls a program from another server that returns data used by the Web application.

## 3 Application overview

The example application uses an Internet API made only for demonstration purposes. When called, the example Internet API returns file names of image files found from a specific folder. These filenames are used when generating the final Web site to add pictures into the page.

### 3.1 Usage

To use the example application, there must be at least two servers available. One or more servers are used as content provider servers and they contain the Internet API application `imagefinder.py`. The MWS is the content consumer server and it contains the mashup application.

#### 3.1.1 Internet API

The Internet API is installed by copying the `imagefinder` folder under the server's documentroot folder. In the MWS environment the documentroot folder is the installation drive:`\Data\Web server\htdocs` folder.

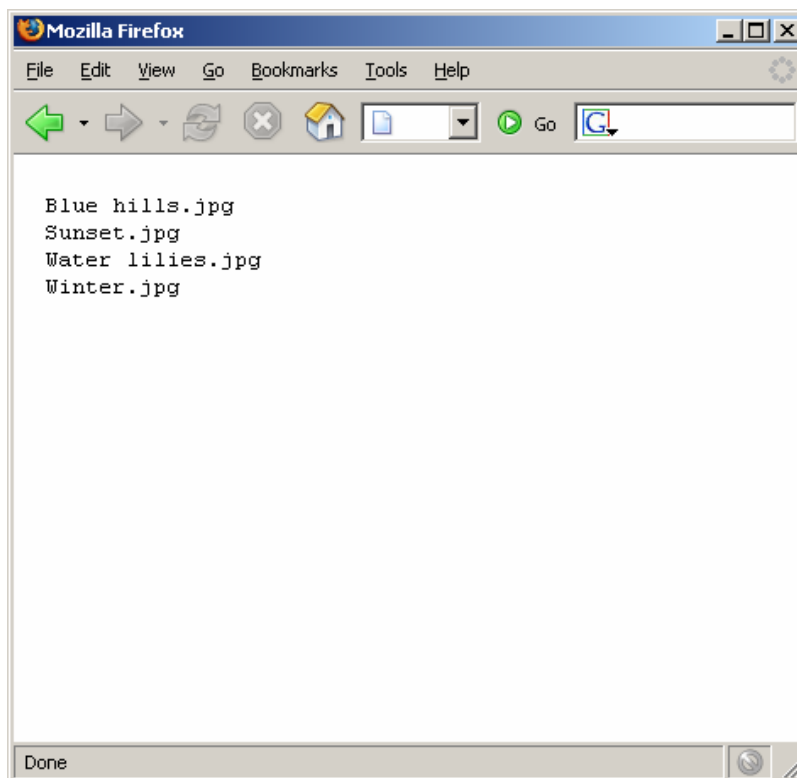


Figure 2: Browser view of data returned from the Internet API

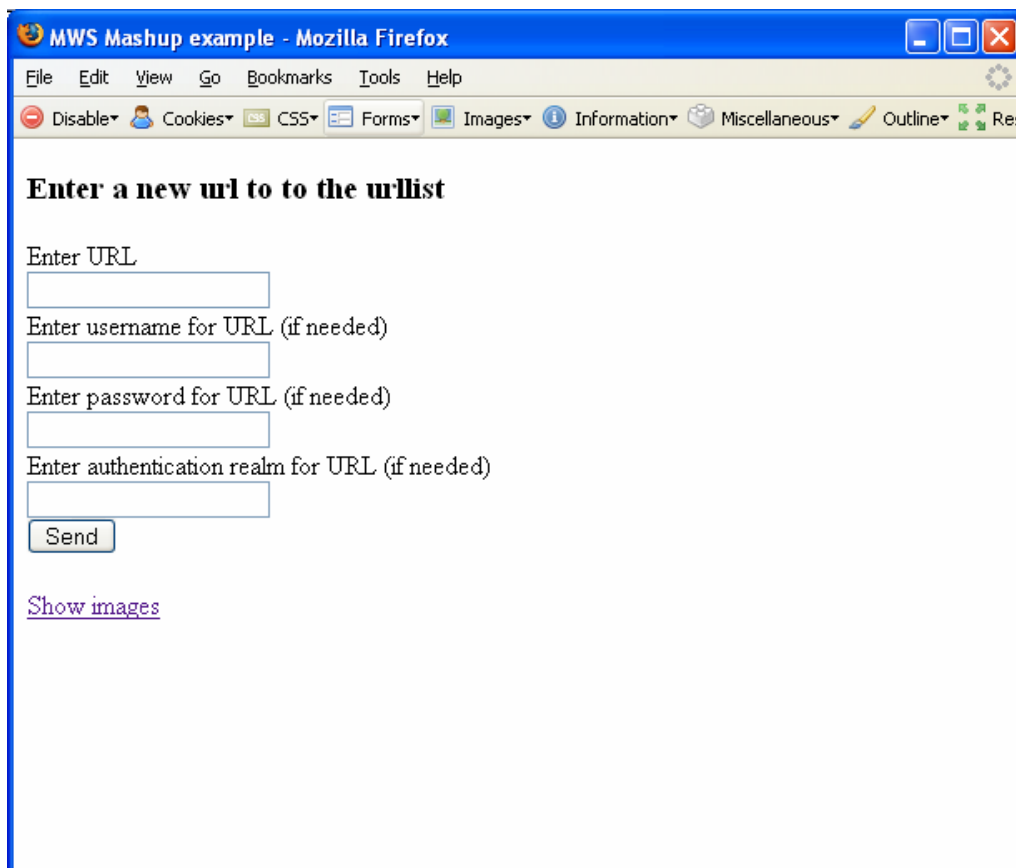
The imagefinder can be tested by giving a browser the server's IP address or URL with the line /imagefinder/imagefinder.py in the end. For example, if the server's IP address is 127.0.0.1 in (W)LAN, the URL would be http://127.0.0.1/imagefinder/imagefinder.py or if the server's URL is http://myphone.mymobilesite.net when connected through the gateway, the URL would be http://myphone.mymobilesite.net/imagefinder/imagefinder.py.

If the server is set up correctly, the browser should return a list of image files found in the imagefinder folder. If the folder is empty, the server returns a blank page.

If the imagefinder folder has access restrictions, the username and password must be given when the Internet API is accessed.

### 3.1.2 Application

To install the mashup application, copy the Mashup folder in the MWS's documentroot folder. After starting the MWS, the mashup application should be working and can be used by browsing to the server's URL and adding the line /Mashup/index.py to the end. For example, if the MWS's IP is 192.168.1.1 in (W)LAN, the URL would be http://192.168.1.1/Mashup/index.py or if the MWS's URL is myphone.mymobilesite.net when connected through the gateway, the URL would be http://myphone.mymobilesite.net/Mashup/index.py.



The screenshot shows a Mozilla Firefox browser window titled "MWS Mashup example - Mozilla Firefox". The browser's address bar is empty. The main content area displays a form with the heading "Enter a new url to to the urllist". The form consists of four text input fields, each with a label: "Enter URL", "Enter username for URL (if needed)", "Enter password for URL (if needed)", and "Enter authentication realm for URL (if needed)". Below these fields is a "Send" button. At the bottom of the form, there is a link labeled "Show images".

Figure 3: Browser view of Mashup application's form

If the application works correctly, the browser should show a form.

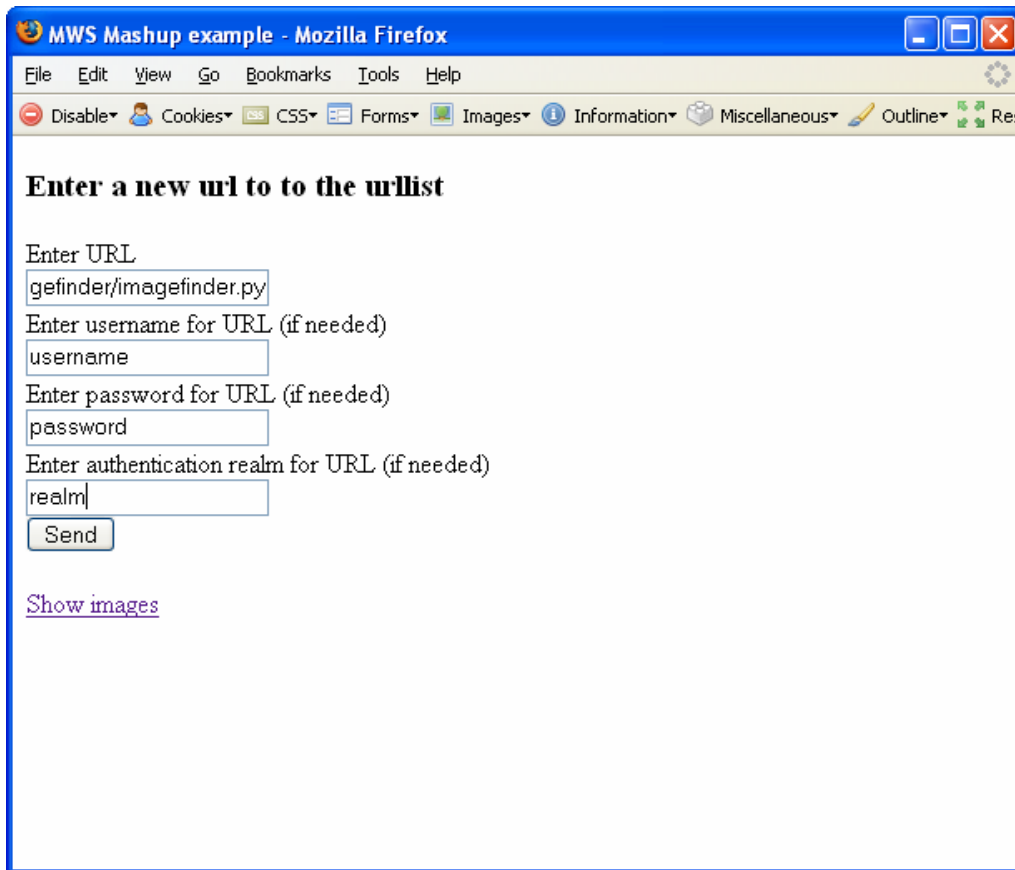


Figure 4: Adding a URL to the form

Add the URL of the imagefinder to the form. This is easiest to accomplish by having copied the URL from the test earlier. The URL should be in the form of `http://ipaddress/imagefinder/imagefinder.py` when connected through (W)LAN or `http://phonename.mymobilesite.net/imagefinder/imagefinder.py` when connected through a gateway. If the content provider server has access restrictions in the imagefinder folder, additional username, password, and authentication realm fields should also be filled. When Send is pressed, the mashup application stores the content provider server's address, username, password, and authentication realm in the mashup folder into a file called URLs. The content of the URLs file can also be viewed and edited directly by using a text editor.

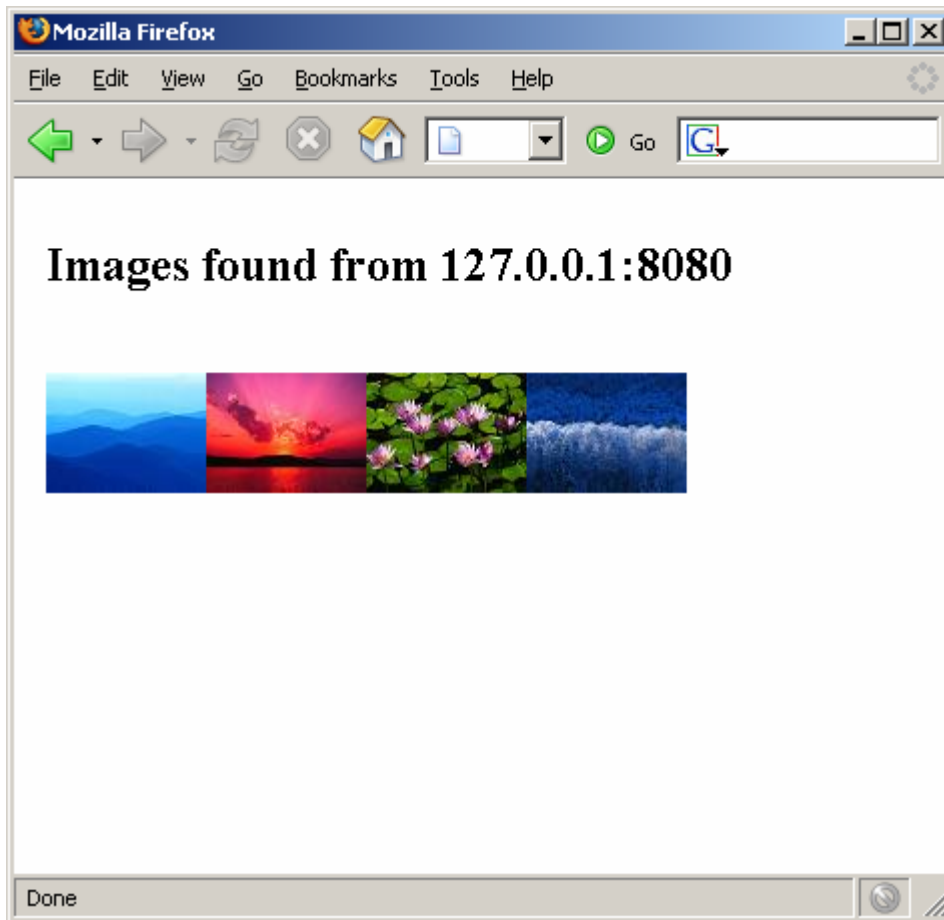


Figure 5: The mashup page generated by the application

After the Show images link has been pressed, the application generates a Web page containing the images found from the content provider servers stored in the mashup application. Note that the phone where the Mashup example is located may display a connection dialog asking about the Internet access point. If this happens, select the same IAP which is already used by MWS.

### 3.1.3 Connections

There are basically two ways to connect to the MWS; (W)LAN and GPRS/3G.

When using (W)LAN, the connection is faster. This reduces loading times, and the IP address is given to a browser to connect.

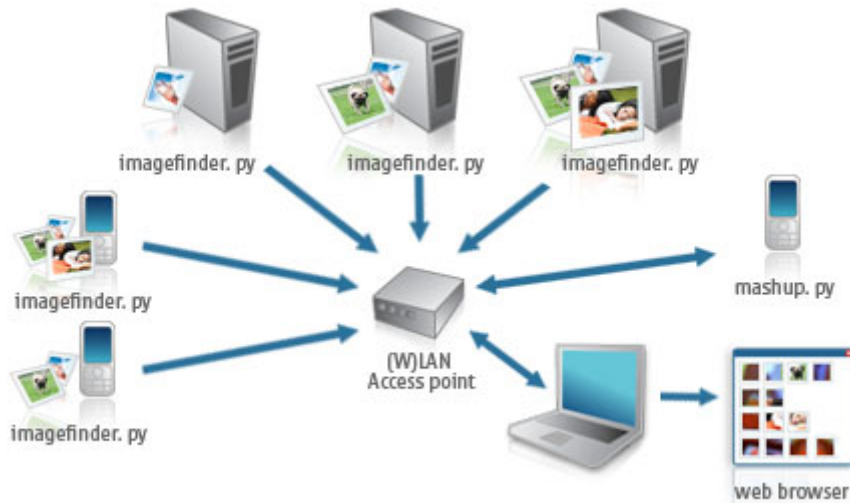


Figure 6: Model of mashup working in a (W)LAN

The GPRS/3G connection uses a gateway which is why the domain name is used to connect a browser to the MWS. This connection is more mobile but due to the high amount of data transfers the mobile operator's contract should be checked to avoid unpleasant surprises, for example, high data transfer costs.

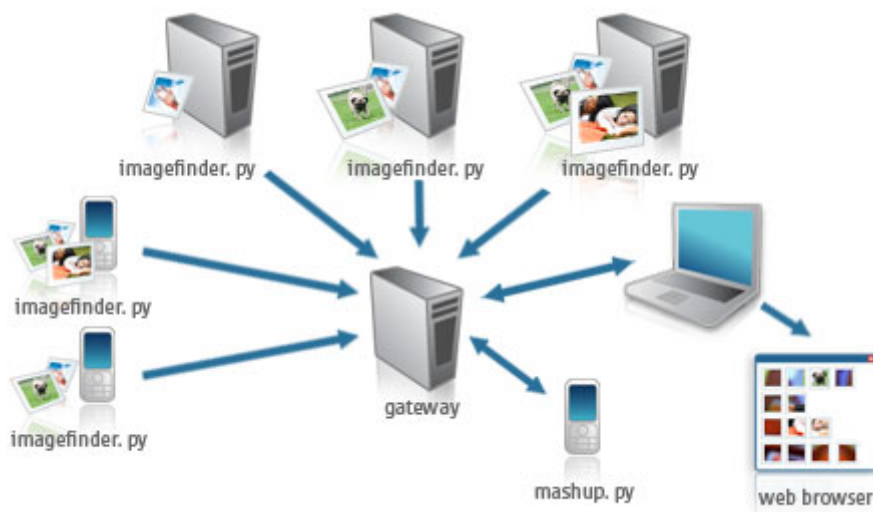


Figure 7: Model of mashup working in a mobile network

## 3.2 Main functions

### 3.2.1 Store URLs

The mashup application can store content provider servers' addresses for future usage in the URLS file.

### 3.2.2 Display images from remote location

The mashup application can generate a Web page that has images from remote locations whose addresses are stored in the URLS file.

### 3.2.3 Find images

When called, the imagefinder Internet API finds and returns the file names of images found in the imagefinder folder.

## 3.3 Class descriptions

| Class     | Description                    | Filename              |
|-----------|--------------------------------|-----------------------|
| HRHandler | Main class of the application  | <i>mashup.py</i>      |
| HRHandler | Main class of the Internet API | <i>imagefinder.py</i> |

## 3.4 Method descriptions

### 3.4.1 handler(req)

Called by the mod\_python for every request. Initializes and calls the HRHandler. Returns the result from `HRHandler.handler()`.

### 3.4.2 handler(self, req)

Handles the request and runs the PSP template. Returns `apache.OK` if no errors occur, returns the error message if something goes wrong. Belongs to the HRHandler class.

### 3.4.3 \_\_init\_\_(self)

Initializes the HRHandler. Belongs to the HRHandler class.

### 3.4.4 parse\_args(self, req)

Parses the data from the http request into a dictionary. Returns the given request. Belongs to the HRHandler class.

### 3.4.5 Walk(self, top)

Finds images from the top folder given as an input. Returns the file names of the found images. Belongs to the HRHandler class.

## 4 Terms and abbreviations

| Term or abbreviation | Meaning   |
|----------------------|---|
| Internet API         | Application Programming Interface for Internet applications |
| MWS                  | Mobile Web Server   |
| URL                  | Uniform Resource Locator                                    |
| IP                   | Internet Protocol   |
| (W)LAN               | (Wireless) Local Area Network                               |
| IAP                  | Internet access point                                       |

## 5 Evaluate this resource

Please spare a moment to help us improve documentation quality and recognize the resources you find most valuable, by [rating this resource](#).