PEKING UNIVERSITY

iMashup Web OS

Developer Guide Manual

https://github.com/sakinijino/iMashup

Daimeng Wang

2010/12/27

Directory

Introduction

iMashup Web OS is an open-source project based on web technology, written in Javascript with dojo toolkit. The aim of the project is to gather services available on the Internet and combine them to achieve a more complex function.

System Requirement

1. Operating System

Since iMashup Web OS is based on web technology and written in Javascript, there is no specific requirement of operating systems.

2. Browser

The dojo toolkit is not so good at compatibility. So we recommend using the 3.x version of Mozilla Firefox browser. It is also recommended that you install the Firebug plug-in in order to acquire more run-time details.

3. Apache

In order to run the internet services, you will need Apache to set up a local server. We recommend you to use the latest 2.2 version of the apache.

Setting up

The following part will introduce you the essential steps to set up iMashup Web OS on your local server.

1. Download Source Code

Download the zip file containing the source code of iMashup Web OS.
Download dojo toolkit archive (version~=1.5.0).

2. Install Apache

Install Apache 2.2.

3. Unpack iMashup

Unpack your downloaded imashup and dojo archives.
Make the directory structure as following

- <base>
  - dijit
  - dojo
  - dojox
  - imashup
  - util

Put the directory into your Apache “httpd” directory or set an alias path.

4. Configure Cross-domain Proxy

Enter the directory where your Apache is located and open “conf/httpd.conf”.

Remove, if any, the "#" character at the front of following sentences:

LoadModule proxy_module modules/mod_proxy.so

LoadModule proxy_ajp_module modules/mod_proxy_ajp.so

LoadModule proxy_connect_module modules/mod_proxy_connect.so

LoadModule proxy_ftp_module modules/mod_proxy_ftp.so

LoadModule proxy_http_module modules/mod_proxy_http.so

Add ProxyPass paths at the end of the file, e.g.:

ProxyPass /googlesearch http://ajax.googleapis.com/ajax/services/search

Save and exit.

And then, enter the directory where your IMashup is located and copy "configs/proxy.js.template" to "configs/proxy.js".

Modify *imashup.configs.proxy.url* to your location.

Add cross-domain URL mappings for your Apache ProxyPass paths

Developer Guide

In this section we are going to analysis the most basic "Text Input" widget and give a brief picture about how to how to develop a component for iMashup Web OS.

Before you go, however, it is highly recommended that you understand the basic of dojo toolkit. See http://www.dojotoolkit.org/.

1. Component Structure

The "Text Input" component contains following files:

imashup/components/widgets/InputPane.js:

Javascript code.

imashup/components/widgets/templates/InputPane.html:

HTML template displayed in the floating pane

imashup/components/widgets/templates/input_large.png:

96*96 icon

imashup/components/widgets/templates/input_small.png:

48*48 icon

2. Javascript Coding

Here is part of the code of InputPane.js:

```
// InputPane.js
dojo.provide("imashup.components.widgets.InputPane");

dojo.require("dijit._Widget");
dojo.require("dijit._Templated");
dojo.require("imashup.core.all");
dojo.require("dojo.io.iframe");
dojo.require("dojo.io.script");

dojo.declare(
```

```
        "imashup.components.widgets.InputPane",
        [dijit._Widget, dijit._Templated],
        {
                imashup_webos_large_icon_url: dojo.moduleUrl(
    "imashup.components.widgets", "templates/input_large.png"),
                imashup_webos_small_icon_url: dojo.moduleUrl(
    "imashup.components.widgets", "templates/input_small.png"),

                templatePath: dojo.moduleUrl("imashup.components.widgets", "templates/InputPane.html"),

                resizable: false,
                maxable: false,
                width: 300,
                height: 60,

                /* Methods
    ……
    */
        }
    );

    imashup.core.componentTypeManager.registerComponentType({
        impl_name : 'imashup.components.widgets.InputPane',
        interface: {
                properties: {},
                methods: {},
                events: {
                        "input": { Function: "oninput", CustomMethod: "/* arguments[0]: String */" }
                }
        },
        mixin_types : ['window']
    });
```

The “dojo.provide(…)” provides an interface for other codes to call the object and functions defined in InputPane.js.

“dojo.require(…)” includes several objects from dojo toolkit.

“dojo.declare(…)” declares the object “imashup.components.widgets.InputPane”, extended from “dijit._Widget” and ”dijit._Templated”.

“imashup_webos_large_icon_url” and “imashup_webos_small_icon_url” is the icon of the widget. To modify the icon, simply replace the "templates/input_large.png" and "templates/input_small.png" with other icons.

“templatePath” defines the HTML template of this widget.

“resizable”, “maxable”, “width”, “height” defines several parameters of the floating pane.

“imashup.core.componentTypeManager.registerConponentType(…)” registers this component to iMashup so it will be shown in the start menu and the user will be able to create new instance of it.

3. HTML Template

Here is the HTML code of the template:

```
<div>
    <div class="inputpane">
        Input here:<input type="text" dojoAttachPoint="inputnode" size="20" style="margin-left:5px;"/>
        <button dojoAttachEvent="onclick:onsubmit">Query</button>
    </div>
</div>
```

Writing a template is just like writing an HTML web page. You can use dojoAttachPoint and dojoAttachEvent property to link certain HTML tag with Javascript function defined in the “InputPane.js”.

4. Mashup Function

Go back to “InputPane.js” and look at the bottom of it:

```
imashup.core.componentTypeManager.registerComponentType({
    impl_name : 'imashup.components.widgets.InputPane',
    interface: {
        properties: {},
        methods: {},
        events: {
            "input": { Function: "oninput", CustomMethod: "/* arguments[0]: String */" }
        }
    },
    mixin_types : ['window']
});
```

In the “imashup.core.componentTypeManager.registerConponentType(…)” function, the “interface.methods” determines which function of the component can be mashed up as receiver, while the “interface.events” decides which function can be mashed up as publisher. Each property in “methods” and “events” is a key-value pair. The key is what the user is supposed to see on the channel interface. The value includes the name of the function and the default Javascript code of the custom method (usually the description of the arguments format).