# Part 1: The Web App folder layout and HTML/JavaScript/CSS:

All web content resources should be put in the WebRoot directory on the server. The layout of the WebRoot directory looks like the following:

WebRoot

|\_\_\_\_\_\_pages

| |\_\_\_\_\_admin

| | |\_\_\_\_abc.html, def.html, ….....

| |\_\_\_login.html

|\_\_\_\_\_\_js

|\_\_\_\_\_\_css

|\_\_\_\_\_\_images

The WebRoot is the highest level of all files which will be on server's web content folder.

Below is the inscription for each folder:

pages: All HTML pages that do not need to be logged in are placed here. Currently there are only one HTML file: login.html. This file is for login page which allows the user to log in. It is directly located under /pages/login.html. The subfolder /pages/admin contains all HTML pages that need to be logged in to access them. Note that all web pages that need admin login should be placed under /pages/admin folder, not directly under the pages folder. In the above example abc.html can only be accessed after the user logs in. However, during the development process, the login function will be disabled fo easy debugging and development. This function is disabled from server side. Client side only needs to treat it as if user has already logged in.

js: This folder contains all JavaScript files. I wrote a userLogin.js to send ajax request to login. Others are not used

css: This folder contains all CSS style sheets.

images: This folder contains all .jpeg, .gif, .png picture files.

Other folders are just some font folders.

You can take a look at the login.html and userLogin.js example to see my code to complete the user login and send data to the server for authorization and authentication.

Note for cross domain ajax request:

When you make cross domain ajax request, instead of using ../adminrest/url/to/request path, you should use the following absolute URL:

[http://www.bluepai.com:8080/webadmin/adminrest/url/to/](http://www.bluepai.com:8080/webadmin/adminrest/url/to/get)request

to make ajax request.

## Part 2: Frontend and Backend communication protocol via RESTful API

Whenever frontend needs to send and receive data from/to backend, RESTful API is used and the data format is always JSON by default unless otherwise specified.

All RESTful web resources are located at the URL path

/adminrest/

which is relative to the web root directory. Note that this URL path is a virtual path that is in parallel with the WebRoot directory. So any request from JavaScript code should use the relative path:

../adminrest/url/to/get

../adminrest/url/topost

This is because all JS files are located under WebRoot/js folder, but /adminrest is at the same level as /WebRoot.

So for example, in userLogin.js, in order to send ajax request to the backend for user authorization, you can see it sends a POST request to the URL:

$.ajax({

url : "../adminrest/login",

data : $("#loginForm").serialize(),

contentType : 'application/x-www-form-urlencoded; charset=utf-8',

type : "POST",

success : **function**() {

window.location.href = '../pages/admin/index.html';

},

error : **function**(jqXHR, status, error) {

alert(jqXHR.responseText);

console.log(error);

console.log(data);

}

});

Here the URL for this ajax request is ../adminrest/login since the JS file is located at /js/userLogin.js and the RESTful resource for login is located at /adminrest/login

The web server at /adminrest/login will do all user authorization and will redirect that to the login.html page upon the user clicks “Logout” button.

The RESTful web service API is provided in the table below. Note that all RESTful web resource with URL path starting with /adminrest needs user login in order to access it. But during development, the user login can be disabled to facilitate development process. Also, not that in the URL given in the table below, they are relative to the WebRoot directory on the server on page 1. So in the JavaScript code, you should use the relative path to send request to the RESTful resources. For example, in the ajax request in the above userLogin.js, I used

url : "../adminrest/login"

The complete RESTful API for communication between frontend and backend is listed below. It is given in this format:

HTTP method: URL

MIME type (content type)

Input data parameters and fields (if applicable in case of GET/POST/PUT request)

Returned data field example

error message (in case of error)

error status code (in case of error)

API usage description

**API #1: Get a full list of all musics**

GET /adminrest/home/playlist

application/json

onSuccess returns JSON array (list of musics): [{"id":10001,"name":"山居岁月.mp3","status":true,"special":false,"playtime":null},{"id":10002,"name":"好天气.mp3","status":true,"special":true,"playtime":"2014-06-01T09:20:50.000-0400"}]

JSON fields:

id: music ID number

name: music name with “.mp3” suffix included

status: This indicates whether the music is in the active playing list or in the inactive staging list. “true” means it is in the active list. “false” means this music is currently in the inactive list (should be marked as red ball in frontend)

special: This indicates whether the music is a special broadcasting item. If true, then the “playtime” must not be null. It must show some datetime information to let the music play at certain time

playtime: This field is not null only if special flag is true. This is the date and time the user wants this music to play. Usually it is the special event broadcasting such as advertisement announcement. The data returned from the server uses the standard UTC+ time zone format.

Usage: This API is used to retrieve all musics currently in the database including both active and inactive items. Frontend can use this information to display all musics as a complete playlist and indicates their status by red and green buttons.

**API #2: Upload a single .mp3 file to the server**

POST /adminrest/home/edit/upload

MIME type: multipart/form-data

onSuccess: return status code 200

Usage: This resource is used when the user wants to upload a local mp3 file to the server. When the user clicks the button “Upload” in the page, there should be a popup window asking him to select the file and then upload. The upload request should be sent to this address. User can only upload the file one by one each time. Everytime the user uploads a file, the frontend JS file should keep the list of uploaded files in an array so that the list will be saved for the final “done” button which will send another ajax request to the POST request below.

**API #3: Finish and finalize all add/remove music to the full playlist**

PUT /adminrest/home/playlist/update

data format sent to the server: application/json

data to send to the server: A JSON array if the same type as that in API #1. It contains the same fields as in #1 above. If the user wants to add the music to the playlist, then “status” field should be 1 (true). If the user wants to remove a music from the playlist, then the “status” field must be marked as 0 (false). Example as below:

[{"id":10001,"name":"山居岁月.mp3","status": 0,"special": 0,"playtime":null},{"id":10002,"name":"好天气.mp3","status":1,"special":1,"playtime":"2014-06-01T09:20:50.000-0400"}]

The above example makes the first entry to be removed from the playlist, while makes the second item to be added to the playlist with playtime information. In this way, the user can add a list of music to active list and also can mark a list of existing items as “inactive” at one button click

onSuccess: It returns the full new list of playlist in JSON array format which is the same as in API #1

onError: It returns HTTP status code 400 (BAD\_REQUEST) and with an error string telling you about the reason

Usage: This is used for the final step after the user finishes uploading all .mp3 files and then check and uncheck the boxes of all music items in the list. If it's checked, then its status is 1. If it's unchecked, then its status is 0. The data sent to the server contains an array of Music objects where “status” field indicates whether this item should be added (1) or removed (0).

**API #4.1: Get all musics of a group**

GET /adminrest/home/playlist/group/{groupName}

return data content type: application/json

This is to get a full list of musics of a group named by the parameter “groupName” in the URL. An example is:

[http://www.bluepai.com:8088/webadmin/adminrest/home/playlist/group/长宁区](http://localhost:8080/webadmin/adminrest/home/playlist/group/长宁区)

The returned data is a JSON array:

[{"id":10001,"name":"山居岁月.mp3","status":true,"special":false,"playtime":null},{"id":10002,"name":"好天气.mp3","status":true,"special":false,"playtime":null},{"id":10003,"name":"Canon.mp3","status":true,"special":false,"playtime":null},{"id":10004,"name":"symphony.mp3","status":true,"special":false,"playtime":null}]

Each element field is the same as those defined in API #1.

OnError: It returns 404 NOT FOUND error code.

**API #4: Add musics to a group**

PUT /adminrest/home/playlist/group/{groupName}/add

application/json

This is an HTTP PUT method. Data sent to the server contain two parts:

groupName in the URL path is a variable. It is the groupName you obtain from the Group list, such as “长宁区组” or “旗舰店组” or “特价组”. You can put this in the URL pathname such as /adminrest/home/playlist/group/长宁区组/add

Message body should contain an array of strings of music names. The music names are those you obtain from the music playlist, such as:

[“山居岁月.mp3”, “好天气.mp3”, “symphony.mp3”]

Note that you must add the suffix .mp3 at the end of the music name, otherwise it will result in error.

OnSuccess: It will return string “添加成功”.

OnError: It will return the HTTP status code 400 (Bad request) and an error string such as “天山路店不能加到该组中”. The error is usually due to the fact that this store already belongs to a group in the same category, therefore cannot be added to the same category group.

Usage: This API will add the list of musics in the array to the specified group.

**API #5: Remove musics from a group**

PUT /adminrest/home/playlist/group/{groupName}/remove

data format sent to the server: application/json

This API is almost the same as #4 exception that you want to remove a list of musics from the group named {groupName}. So everything is the same as in #4.

OnSuccess: It will return string “删除成功”.

OnError: It will return the HTTP status code 400 (Bad request) and an error string such as “天山路店不能从该组删除”.

**API #6： Monitor the Android device status**

GET /adminrest/musicdevice/monitor

MIME type: application/json

returned data example:

[{"deviceID":14230739172120,"connectStatus":false,"timeReported":"2014-05-28T09:28:45.000-0400","storeName":"长宁店"},{"deviceID":279317396357982,"connectStatus":false,"timeReported":"2014-06-03T21:30:20.000-0400","storeName":"浦东一号店"},{"deviceID":354659215,"connectStatus":false,"timeReported":"2014-07-01T12:00:00.000-0400","storeName":"人民广场店"}]

In the above JSON array

deviceID: Android device ID

connectStatus: whether this device is currently connected (true or false)

timeReported: last heartbeat signal received from this device

storeName: the corresponding store name of this Android device

Usage: data returned from the server can be used to show the device status (connected or disconnected)

**API #7: Web Admin Login**

POST /adminrest/login

sent data type: application/x-www-form-urlencoded

Input data: There are two form parameters:

1. username

2. password

These two should be set in the HTML form as its “name” attribute.

OnSuccess: After the user successfully logs in the account, it will be directed to the Web Admin homepage which is in the URL:

/pages/admin/index.html (relative URL)

or full URL:

<http://www.example.com:8080/webadmin/pages/admin/index.html>

onError: It will return error message string “Incorrect username or password” to the client side.

Usage: The Web Admin Login web page should be located at /page/login.html (Relative URL to the web context root). This HTML page should contain form element to have “username” and “password” fields in the web form.

Note: Any HTML pages under /pages/admin/ subpath needs user authentication to see them. So all web admin HTML pages should be placed under this subpath like

/pages/admin/index.html

/pages/admin/table.html

…...

However, during development stage, this login functionality is disabled for easy debugging.

**API #8: Web Admin Logout**

GET /adminrest/logout

onSuccess: This will logout the current webadmin user and after logout it should be redirected to the login page which is:

/pages/login.html

So after logging out, the user will still go back to the login page.

**API #9: Get a list of all groups**

GET /adminrest/home/groups

returned data type: application/json

No input parameters requried

Returned data in JSON array. Example:

[{"id":1000,"groupName":"长宁区","category":"区域","expireDate":null},{"id":1001,"groupName":"旗舰店","category":"级别","expireDate":null},{"id":1002,"groupName":"静安区","category":"区域","expireDate":null},{"id":1004,"groupName":"浦东新区","category":"区域","expireDate":null},{"id":1005,"groupName":"徐汇区","category":"区域","expireDate":null},{"id":1007,"groupName":"打折组","category":"临时","expireDate":"2014-07-24T21:09:34.000-0400"},{"id":1008,"groupName":"促销组","category":"临时","expireDate":"2014-07-24T21:19:23.000-0400"},{"id":1017,"groupName":"崇明区","category":"区域","expireDate":null}]

JSON fields:

id: group ID

groupName: group name

category: group category, can be one of the three: 区域，级别，临时

expireDate: The expiration date for 临时 category group since this group category will expire after certain date and time which is set by the user. Usually this is null. But if it's not null, then it will have something like

"expireDate":"2014-07-24T21:09:34.000-0400"

**API #10: Get group by category**

GET /adminrest/home/groups/category/{category}

returned data type: application/json

URL path parameter: The category you want to query from, e.g., 区域，级别，临时. So for example, if you use the following URL:

/adminrest/home/groups/category/区域

You will get the following JSON response:

[{"id":1000,"groupName":"长宁区","category":"区域","expireDate":null},{"id":1002,"groupName":"静安区","category":"区域","expireDate":null},{"id":1004,"groupName":"浦东新区","category":"区域","expireDate":null},{"id":1005,"groupName":"徐汇区","category":"区域","expireDate":null},{"id":1017,"groupName":"崇明区","category":"区域","expireDate":null}]

The above JSON array contains all groups in the 区域 category.

This API is used to retrieve all groups in a certain category. This is useful for frontend to display a full list of groups in a category when user selects a certain category.

**API #11: Get group by group ID**

GET /adminrest/home/groups/id/{id}

return data: application/json

URL path parameter: group ID that is used to retrieve a group by ID, for example, if you use the following URL

/adminrest/home/groups/id/1000

You will get the group with id=1000 in JSON:

{"id":1000,"groupName":"长宁区","category":"区域","expireDate":null}

onError: A HTTP status code 404 NOT FOUND will be returned

**API #12: Create a new group**

POST /adminrest/home/groups/create

data format sent to the server: application/x-www-form-urlencoded

Input data parameters: Form data parameters

1. groupName: This is the group name of the group such as “便利店组”

2. groupCategory: group category from one of the three: 区域，级别，临时

3. expireDate: expiration date and time for a group in 临时 category. It should be null for 区域，级别. But it should be not null for 临时 category.

OnSuccess: It will return message: 创建新组成功

onError: If the expireDate is null for 临时, it will return error message: 临时组过期日期不能为空

If the group is already created in the database, it will return error message: 不能创建新组(组名不能重复)

Use this API to create a new group using form elements.

**API #13: Delete group by ID**

DELETE /adminrest/home/groups/id/{id}

No date sent to the server. But the path parameter id is the group ID. So for example sending DELETE ajax to the following URL:

/adminrest/home/groups/id/1000

will delete group with id=1000.

**API #14: Add a list of stores to a group**

POST /adminrest/home/store/add/to/group

data type sent to the server: application/x-www-form-urlencoded (Form Data)

Input parameters:

storeName: An array containing all store names that you want to add to the group. Example: [“重庆路店”， “天山路店”， “浦东一号店”]

groupName: The group name which you want to add to. Example: “长宁区组”

onSuccess: It will return 添加成功 message

onError: It will return 门店A不能放到长宁区组 error message. This is usually due to the two reasons:

1. Store A is already in group B

2. Store A already belong to another group which is in the same category as group B.

This API is used to batch add a list of stores to a group. When user check the stores using check marks, and then click the arrow button → to a group, it will call this API to add all stores to the group.

**API #15: Remove a list of stores from a group**

PUT /adminrest/home/store/remove/from/group

data type sent to the server: application/x-www-form-urlencoded (Form Data)

This is the reverse of API #14. This will remove a list of stores from a group.

Input parameters:

storeName: An array containing all store names that you want to add to the group. Example: [“重庆路店”， “天山路店”， “浦东一号店”]

groupName: The group name which you want to add to. Example: “长宁区组”

onSuccess: It will return 删除成功 message

onError: It will return 门店A不能从长宁区组删除 error message

This API is used to batch remove a list of stores to a group. When user check the stores using check marks, and then click the arrow button → to a group, it will call this API to remove all stores to the group.

**API #16: Get all unassigned stores**

GET /adminrest/home/store/unassigned

returned data type: application/json

This is to get a list of all stores that are not assigned to a group yet. On successful return, it will return a JSON array of store objects. For example:

[{"storeID":"C11022","storeName":"重庆北路店","deviceID":10436312250000},{"storeID":"C11024","storeName":"山海关路店","deviceID":0}]

fields:

storeID: String

storeName: String

deviceID: long, the device ID of the device in this store

If return on error, it will return status code 404 (NOT FOUND)

**API #17: Get all stores in a group**

GET /adminrest/home/store/{groupName}

return data type: application/json

This is to get all stores in a certain group. The URL path parameter groupName is given as a parameter for group name. It must be the same name as what is returned by calling API #9, e.g., 长宁区. So an example of request will be like:

GET /adminrest/home/store/长宁区

On successful return, it will return a JSON array:

[{"storeID":"12345","storeName":"长宁店","deviceID":14230739172120},{"storeID":"43209","storeName":"浦东一号店","deviceID":279317396357982},{"storeID":"C11019","storeName":"人民广场店","deviceID":354659215}]

So the above are all three stores in the group “长宁区”.

On error, it will return 404 (NOT FOUND).