Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Institute of Information Systems
Prof. Moira C. Norrie
Amir Tafreshi, Linda Di Geronimo

Mobile and Personal Information Systems

Autumn 2015

Exercise 2

Inspired by the Hello.wall project presented in the lecture, we ask you to develop an application that displays information using such a wall. In this exercise we use a web-based ambient wall framework that caters for the management of the data as well as the data display using a web-based wall implementation.

Your task is to design and implement an application that generates data which can then be displayed using the ambient wall framework. The project should be presented during the exercise session on 05.11.2015.

Note that this exercise involves using an ambient wall to display live awareness information from Twitter. If you do not have a Twitter account and do not want to create one, you can either contact us to provide you an authentication key for your access to Twitter API or you could alternatively develop a mock-up where WordPress posts are used to simulate social media posts.

Exercise 2.1 *Scenario* As a first step, you will define a scenario and the type of data to be displayed. In the lecture, we have seen an example where the activity level of followed users of a user in Twitter is displayed using the ambient wall. In this exercise we ask you to come up with an application that visualizes live awareness information about followers/tweets, etc. from Twitter which addresses the issue of scalability for users who are following hundreds of users. Your application should be based on your ideas developed in the breakout session in week 5, but you are also free to refine or change your application based on feedback and proposals presented by other groups.

Exercise 2.2 *Framework Setup* As already mentioned, we will use an ambient wall framework for managing and displaying the data. The framework is integrated into the ScreenPress platform, enabling WordPress and functionality to be used, and furthermore, allowing development of cross-device ambient walls (e.g dividing the snapshots and representing each one on a different device (i.e. different web-based walls, see Fig. 1).

To integrate the framework into your already installed ScreenPress, first, download and extract the framework from the course web site (`https://globis.ethz.ch/files/2015/10/Wall.zip`). Afterwards, copy and update the files as follows:

- Copy the new given files in "ScreenPress_Node" folder into the same folder in your already installed ScreenPress. Then, Find the following lines in the corresponding files and update them according to your hostname and WordPress folder :
  - **wall.js**:
    - `var mainUrl = "http://localhost/WordPressfolder/";`
  - **visualisation.html**:
    - `<script src="http://localhost/WordPressfolder/wp-includes/js/raphael.js" type="text/javascript" charset="utf-8"></script>`
    - `<script src="http://localhost/WordPressfolder/wp-includes/js/jquery.js" type="text/javascript" charset="utf-8"></script>`
    - `var url = "http://localhost";`

- Copy the given "WordPress" files into your already installed WordPress. Then, go to the "Pages" menu in the WordPress dashboard and add a new page ("Add New") with title "wall" and set its Template to "Wall JSON OUTPUT".
- You should get the sample Twitter application presented in the lecture running. To do so, you would need to create your own Twitter Oauth on `https://apps.twitter.com` and accordingly update the *TwitterOauth.php* file located in "Wordpress\wp-content\themes\twentyfourteen". You are free to adapt the app to your idea using the Twitter REST APIs [1]. *please note that each app Oauth in Twitter has the limitation*[2] *of 180 requests per 15 minutes.*

The framework consists of three main files: (*1*) The wall.php in WordPress handles the data generation and abstraction, where the information to be displayed is in json format. (*2*) The visualisation.html is built for representation of the abstract data as a wall of circles. To do so, a compatible cross-browser javascript library called Raphael[3] is used for drawing graphical circles. (*3*) The wall.js in Node.js is responsible for retrieving the data from WordPress at a configurable fixed time interval and distributing the retrieved data across a single or multiple web walls (Fig. 1) .

To change the time interval, you can modify the following code in "wall.js" file.

```
setInterval(function()
   updateWall();
, 1000* 21); // 1000 * 1 = 1 second
```

Figure 1: Representation of the distributed input data across multiple displays

Exercise 2.3 *Run your Application* In order to run the application, you first need to start the node server as follows:

(*1*) Open the command prompt and go to the *Wall_Node* folder.

(*2*) Run the *wall.js* by executing the following command:

```
node wall
```

Then, you can open the application on a browser, using the following link according to your host name and your application name:

```
http://HostName:5000/wall/?display=appName
```

You can also see the output of the aforementioned sample generated input data on the wall using the following links:

- `http://HostName:5000/wall/?display=sample`
- `http://HostName:5000/wall/?display=static`
- `http://HostName:5000/wall/?display=random`
- `http://HostName:5000/wall/?display=twitterFriends`
- `http://HostName:5000/wall/?display=twitterOverall`

[1] `https://dev.twitter.com/rest/public`
[2] `https://dev.twitter.com/rest/reference/get/application/rate_limit_status`
[3] `http://raphaeljs.com`

Exercise 2.4 *Implementation* Having the application scenario defined, you can now proceed to implement the application. You can implement your application by modifying the aforementioned wall.php file to generate input data for the ambient wall framework. Below, you can see a main snippet of the code that illustrates what your application has to generate. Fig. 2 shows the output of this code:

```
//Initialize wall size
$temp["width"] = 500;
$temp["height"] = 200;
$temp["leftgutter"] = 300;
$temp["bottomgutter"] = 100;

//Initialize wall axis
$temp["axisx"] = array("", "", "", "", "");
$temp["axisy"] = array("", "");

// Circle positions, sizes (0 to 1), colors (-1 to 1) and labels
$temp["xs"] = array(0, 0, 0, 0, 0, 1, 1, 1, 1, 1);
$temp["ys"] = array(0, 1, 2, 3, 4, 0, 1, 2, 3, 4);
$temp["size"] = array(0.5, 1, 0.5, 1, 0.5, 1, 0.5, 1, 0.5, 1);
$temp["color"] = array(0, 0.1, 0.2, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9, 1);
$temp["label"] = array("0,0", "0,1", "0,2", "0,3", "0,4",
 "1,0", "1,1", "1,2", "1,3", "1,4");
...
```

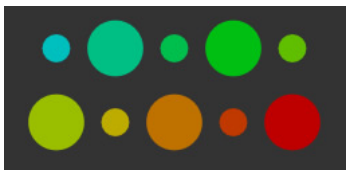

Figure 2: Sample circles on the wall

To try out some sample generated input data, you can run the following urls in your web browser:

- // Input data of Fig. 2
  http://HostName/WPfolder/wall/?ptype=sample
- // a 8X7 wall with equivalent circle size, color and label
  http://HostName/WPfolder/wall/?ptype=static
- // a 10 * 7 wall which generates circles with random sizes and colors
  http://HostName/WPfolder/wall/?ptype=random