Tutorial 2: Show list of Posts on App home screen

The previous tutorial we have added a new menu item to the main menu of the mobile app. However, nothing happens when tapping on the Posts menu. This tutorial will guide you:

- Create new screens for Post app
- Display a list of posts when tapping on the Posts menu.

Step 1: Create the primary app Resource

Each app requires a primary *Resource* to control the app's settings. In this case, we will create ***PostResource*** class as the primary Resource. Create **Post Resource.php** file in **Resource** folder and add the following sample code

```
<?php
namespace Apps\Posts\Api\Resource;
use Apps\Core_MobileApi\Api\Resource\ResourceBase;
class PostResource extends ResourceBase
{
        public $resource_name = "post";
        public $module_name = "post";
        public $title;
        public $description;
        public $text;
}
```

In the sample code above, we created **PostResource** class extended from **ResourceBase** class. 2 properties **$resource_name** and **$module_name** hav e to be unique and different from other apps. All other class's properties are corresponding to database fields of Posts object

Step 2: Implement resource API service

In **Service** folder, create **PostApi** class what is extended from **AbstractResourceApi** class and implements **MobileAppSettingInterface** interface.

At this time, we will implement two main methods of **PostApi** class

- Method **PostApi::findAll()** is to get all posts.
- Two method **PostApi::getAppSetting()** and **PostApi::getScreenSetting()** are to register settings of the Post app to the Native Mobile App. Minimal code show as below:

PostApi.php

```php
<?php

namespace Apps\Posts\Api\Service;

use Apps\Core_MobileApi\Adapter\MobileApp\MobileApp;
use Apps\Core_MobileApi\Adapter\MobileApp\MobileAppSettingInterface;
use Apps\Core_MobileApi\Api\AbstractResourceApi;
use Apps\Posts\Api\Resource\PostResource;

class PostApi extends AbstractResourceApi implements MobileAppSettingInterface
{
        public function findAll($params = [])
        {
                $posts = [
                        new PostResource([
                                'post_id' => 1,
                                'title' => "Post's title example",
                                'description' => "Post's description example",
                                'text' => "Post's description example"
                        ])
                ];
                return $this->success($posts);
        }

        public function getAppSetting($param)
        {
                $app = new MobileApp('post' ,[
                        'title'=> 'Posts',
                        'home_view'=>'menu',
                        'main_resource'=> new PostResource([])
                ]);
                return $app;
        }

        public function getScreenSetting($param)
        {
                $screenSetting = new ScreenSetting('post', [
            'name' => 'posts'
        ]);
                $resourceName = PostResource::populate([])->getResourceName();
                $screenSetting->addSetting($resourceName, ScreenSetting::MODULE_HOME);
                $screenSetting->addSetting($resourceName, ScreenSetting::MODULE_LISTING);
                $screenSetting->addSetting($resourceName, ScreenSetting::MODULE_DETAIL);

                return $screenSetting;
        }

/* Keep other methods empty for implementation later */
}
```

We have finished implementing two primary classes of **Posts** app. Now we need to register **Posts** app to the Phpfox system

Step 3: Register Post App to Phpfox System

Create new ***hooks mobile_api_routing_registration.php*** in **hooks** folder with the following code

mobile_api_routing_registration.php

```
<?php
/**
Define RestAPI services
*/
$this->apiNames['mobile.post_api'] = \Apps\Posts\Api\Service\PostApi::class;
/**
Register Resource Name, This help auto generate routing for the resource
Note: resource name must be mapped correctly to resource api
*/
$this->resourceNames['post'] = 'mobile.post_api';
```

Register **PostApi** service in ***start.php*** of the **Posts** app
PostApi.php

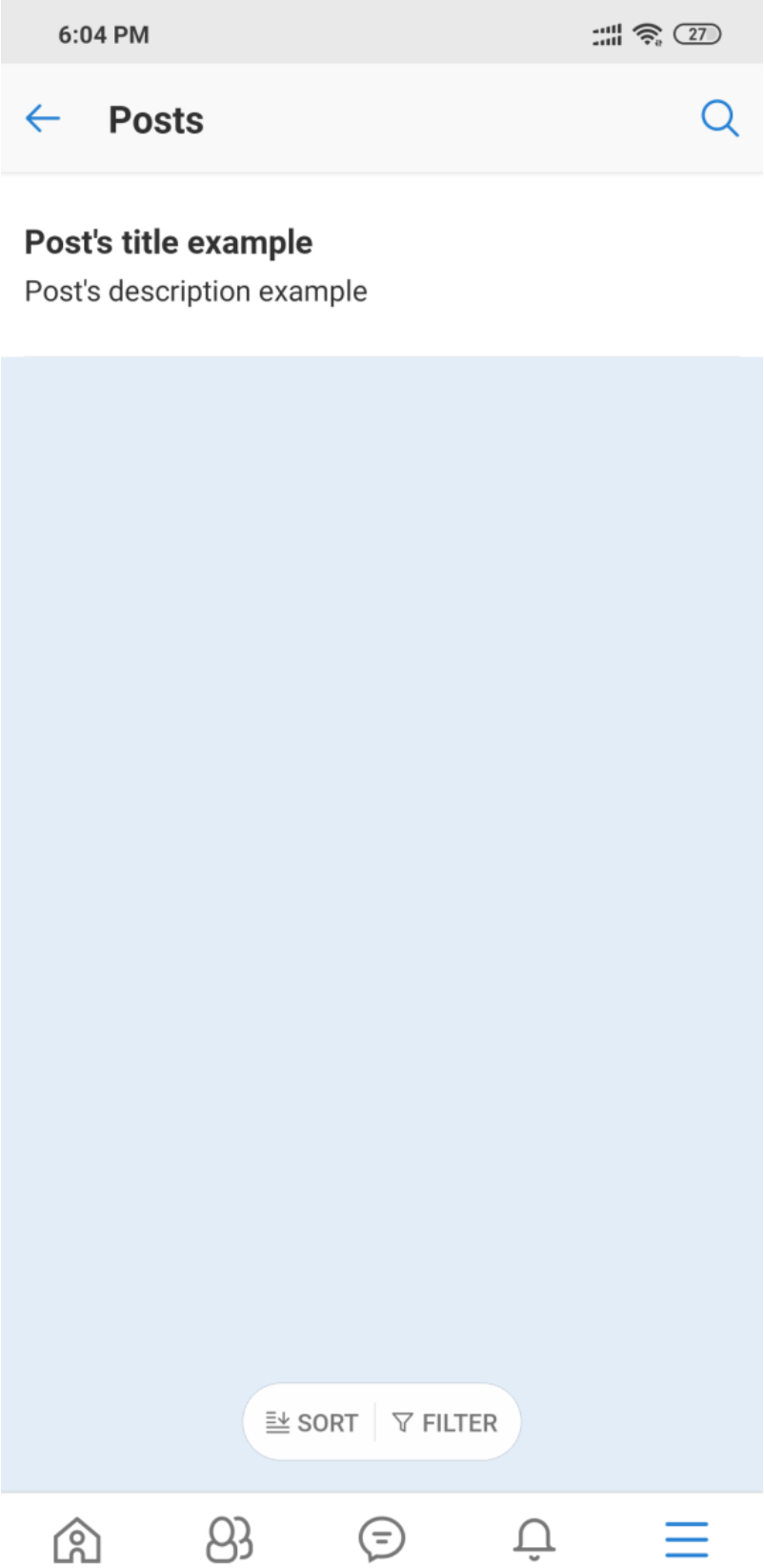
```
<?php
namespace Apps\Posts;
use Phpfox;

Phpfox::getLib('module')->addServiceNames([
        'mobile.post_api' => Api\Service\PostApi::class,
]);
```

Step 4: Test the integration

A new app screen on the mobile app was registered and connected with the **Posts** menu after all steps above are completed. The new app screen called main Post's app screen, and it combines with API function to get data of all posts via RESTful API and show results on the screen

Now, you can clear cache in AdminCP and re-open the Native Mobile App, click on the Posts menu item to see how it works.



Let's review the source structure of Posts app

We put all source code to implement API integration in the **Api** folder