Mobile API Design Guide

Restful Mobile API design & Implementation

On this section, we will go through RESTful API design concepts, how to use and extend.

> **Requirement**
>
> 2 following phpFox apps are required:
>
> - RESTful API app (version 4.2.2 or later)
> - Mobile API app (version 4.2.0 or later)

RESTful API design concepts & convention

The fundamental concept in any RESTful API is the **Resource**. A **Resource** is an object with a type, associated data, relationships to other Resource objects, and a set of methods.
RESTful API naming convention follows resource-based way and supports standard HTTP GET, POST, PUT and DELETE methods.

In this article, we will use the **Blog** app as an example.

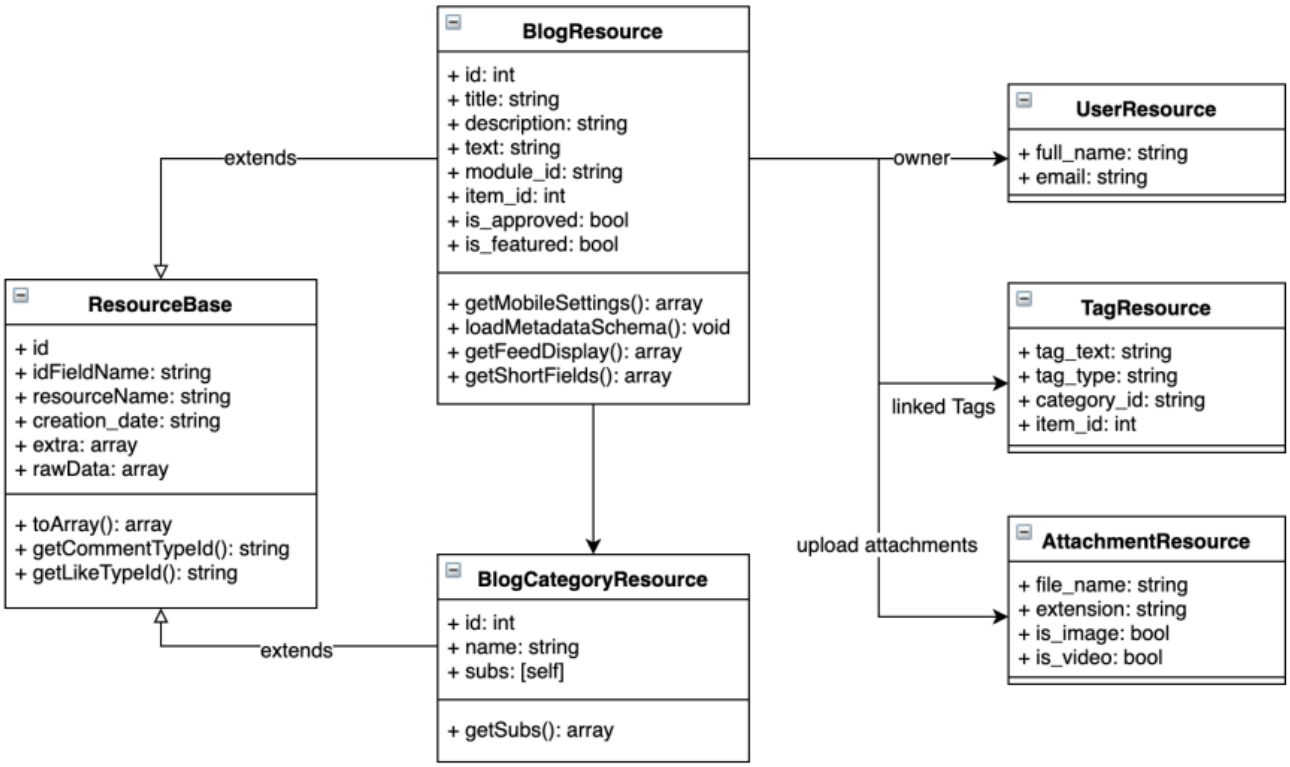
Resources

Blog app we have defined 2 resources:

- *"blog" resource* refers to a Blog entry
- *"blog-category" resource* refers to Blog Category entry
- *"blog" resource* has a many-to-many relationship with *"blog-category" resource*

Resource Relationship

A **Resource** can have either one-to-many or many-to-many relationships with other Resources using property. The Class diagram below shows the relationship of resources in the ***Blog*** app



API to access Blog resource

Each resource has the following standard APIs:

- Listing & search resource
  - Route: "/blog"
  - Method: "GET"
- Create new resource
  - Route: "/blog"
  - Method: "POST"
  - Request data in JSON format
- Update resource
  - Route: "/blog/:id"
  - Method: "PUT"
  - Request data in JSON format
- Get Form structure for creating/updating
  - Route: "/blog/form/" +
  - Method: "GET"
  - Parameters: use "id" param for edit case
- Delete resource
  - Route: "/blog/:id"
  - Method: "DELETE"
- Make as featured
  - Route: "/blog/feature/:id
  - Method: "PUT"
- Make as sponsor
  - Route: "/blog/sponsor/:id
  - Method: "PUT"

Blog Category resource and other resources also have similar APIs

Resource response in JSON

A JSON presentation of a resource looks like an example below.
Blog Resource Response Example


```
{
        "status": "success",
        "data": {
                "resource_name": "blog",
                "module_name": "blog",
                "title": "Et a dolor eum libero nostrum cumque.",
                "description": "Esse beatae voluptas officiis...",
                "module_id": "blog",
                "item_id": 0,
                "is_approved": true,
                "is_sponsor": false,
                "is_featured": false,
                "is_liked": false,
                "is_friend": false,
                "is_pending": false,
                "post_status": 1,
                "text": "Esse beatae voluptas officiis ratione ",
                "image": null,
                "statistic": {
                        "total_like": 0,
                        "total_comment": 0,
                        "total_view": 1,
                        "total_attachment": 0
                },
                "privacy": 0,
                "user": {
                        "full_name": "Sheridan Hahn",
                        "avatar": null,
                        "id": 841
                },
                "categories": [
                        {
                                "id": 4,
                                "name": "Family & Home",
                                "subs": null
                        },
                        {
                                "id": 9,
                                "name": "Sports",
```

```
                                "subs": null
                        }
                ],
                "tags": [
                        {
                                "tag_text": "tag me",
                                "id": 3
                        }
                ],
                "attachments": [],
                "id": 1,
                "creation_date": "2016-06-21T04:53:48+00:00",
                "modification_date": null,
                "link": "http://localhost:7788/blog/1/et-a-dolor-eum-libero-nostrum-cumque/",
                "extra": {
                        "can_view": true,
                        "can_like": true,
                        "can_share": true,
                        "can_delete": true,
                        "can_report": true,
                        "can_add": true,
                        "can_edit": true,
                        "can_comment": true,
                        "can_publish": false,
                        "can_feature": true,
                        "can_approve": false,
                        "can_sponsor": true,
                        "can_sponsor_in_feed": false,
                        "can_purchase_sponsor": true
                },
                "self": null,
                "links": {
                        "likes": {
                                "ref": "mobile/like?item_type=blog&item_id=1"
                        },
                        "comments": {
                                "ref": "mobile/comment?item_type=blog&item_id=1"
                        }
                },
                "feed_param": {
                        "item_id": 1,
                        "comment_type_id": "blog",
                        "total_comment": 0,
                        "like_type_id": "blog",
                        "total_like": 0,
                        "feed_title": "Et a dolor eum libero nostrum cumque.",
                        "feed_link": "http://localhost:7788/blog/1/et-a-dolor-eum-libero-nostrum-cumque/",
                        "feed_is_liked": false,
                        "feed_is_friend": false,
                        "report_module": "blog"
                }
        },
        "error": null
}
```


Control Resource API Response

In the new Core Mobile API app, we define **ResourceBase** class for controlling resource output, mapping data fields and generating routes for Resource API.
Every new Resource class should extend from this base class and use the list of core resources for its relationship when building APIs. This way will help to reduce the code to build the API

Reusable core resources and common use cases

All resource classes are defined in the folder "*PF.Site/Apps/core-mobile*-api/Api*/Resource*" under name Space "*\Apps\Core_MobileApi\Api\Resource*"

Core's Resource	Description	Use Case

UserResource	Presentation of a Phpfox's User	• Use for post's Author
TagResource	Tag feature in Phpfox	• List of tags of a post (blog, event, listing)
NotificationResource	Core notification feature	• Reuse notification feature for other resources
LikeResource	Core like feature	• Reuse like feature for other resources
CommentResource	Core comment feature	• Reuse comment feature for another resource
FriendResource		
FileResource		
FeedResource		
AttachmentResource		

Reusable Core Objects

Unlike resource, Core Object is used to group related properties of the Resource into an object. List of reusable objects are listed in the following table:

Object Class	Use Case
Image	Presentation as an Image of an Item (blog, marketplace listing)
Privacy	Privacy of an item
FeedParam	Feed parameter of an Item
Statistic	Total Like, Total Comment, Total View of an Item

Mapping property, related resource and object

To create a Resource object, we can fetch data from the database as an Array and then populate to the Resource. Below is the sample code to creating a Resource from PHP Array

```
<?php
// Fetch blog data from database then create blog resource
        $blog = \Phpfox::getService("blog")->get($blogId);
        $blogResource = new BlogResource($blog)
```

We define some of convention way to mapping from a data source to resource's properties

Map resource's properties by naming convention:
If the data source has key same with the property name. It's auto mapped

Map with User Resource
If the data source has a set of **"user_"** prefix key and `user` property, It auto combine field to UserResource and Map to user property of the current resource

Manual mapping
We can override or manual mapping fields use Setter and Getter methods. Setter method uses to control input data, Getter control the output. The naming convention of setter & getter methods follow examples bellow

Resource's property name	Setter method	Getter method
name	setName()	getName()
short_description	setShortDescription()	getShortDescription()

is_approved	setIsApproved()	getIsApproved()

All fields query from the database table has a String data type. But Native App requires to return exactly data type in JSON.
Override **"loadMetadataSchema"** method of resource to control the response data type.

Following is an example:

```
<?php

/* A lot of code above */
class PostResource extends ResourceBase
{
/* ... */
        protected function loadMetadataSchema(ResourceMetadata $metadata = null)
        {
                parent::loadMetadataSchema($metadata);
                $this->metadata
                    ->mapField('title', ['type' => ResourceMetadata::STRING])
                    ->mapField('description', ['type' => ResourceMetadata::STRING])
                    ->mapField('item_id', ['type' => ResourceMetadata::INTEGER])
                    ->mapField('module_id', ['type' => ResourceMetadata::STRING])
                    ->mapField('is_approved', ['type' => ResourceMetadata::BOOL])
                    ->mapField('is_sponsor', ['type' => ResourceMetadata::BOOL])
                    ->mapField('is_featured', ['type' => ResourceMetadata::BOOL])
                    ->mapField('is_liked', ['type' => ResourceMetadata::BOOL])
                    ->mapField('is_friend', ['type' => ResourceMetadata::BOOL])
                    ->mapField('post_status', ['type' => ResourceMetadata::INTEGER]);
        }
}
```

Develop restful resource APIs

Following section, we base on an app named **Posts** clone from the Core Blog app to write example code.
The features of Posts app are the same as the Blog app, just change routing from "blog" to "post". Features included:

1. Posts listing, search, my posts, friend's post, browse by category
2. Create or Edit a Post
3. Delete a Post
4. Categories
5. Tags to a Post

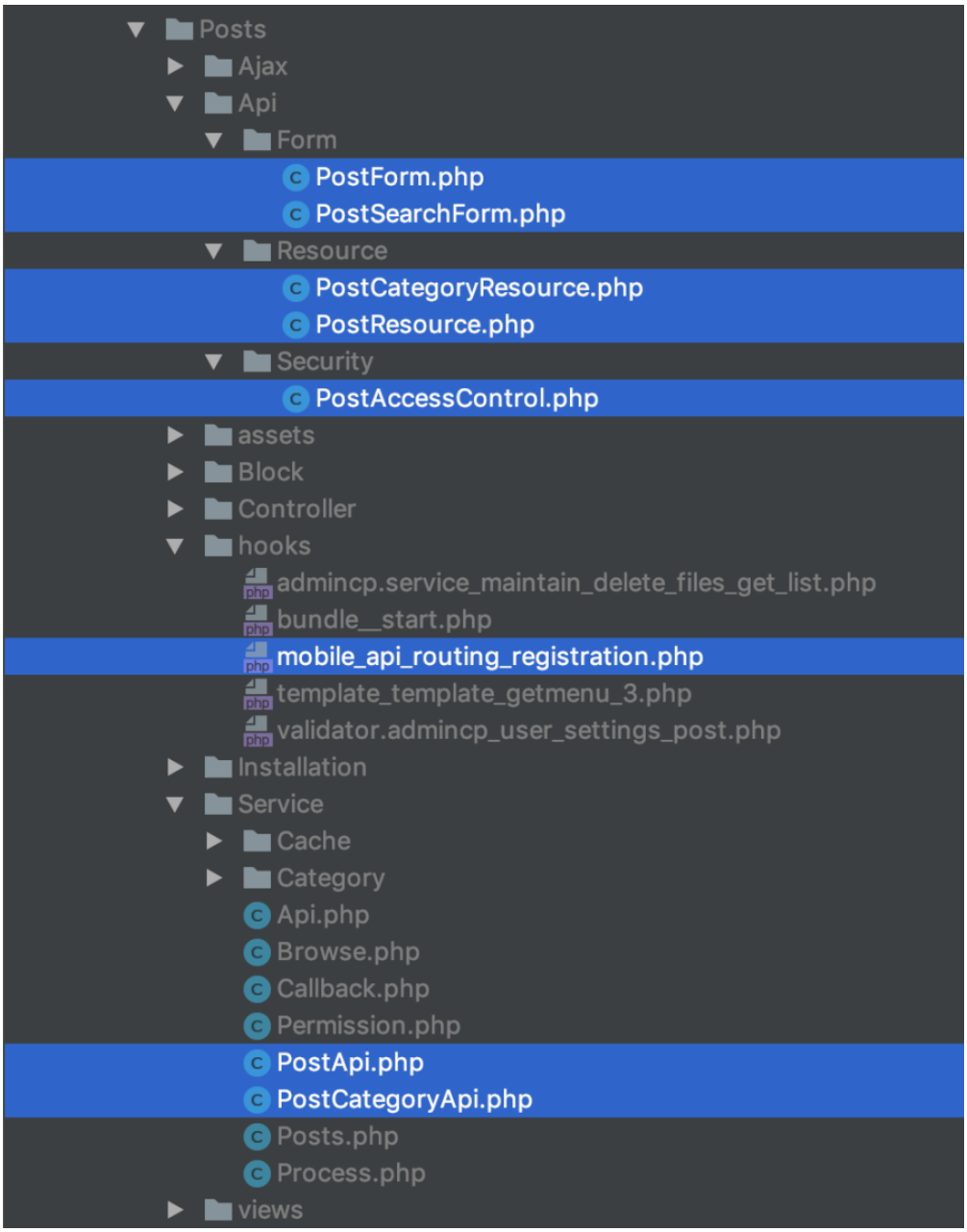
Posts app information:

- App's ID: "Posts"
- App's Dir: "PF.Site/Apps/Posts"
- App's Alias: "post"
- App's Routing: "post/*"
- App's Namespace: "\Apps\Posts"

Step by step to extend APIs:

1. Create a resource class for each resource of the App
2. Create a Service to handle API request for each resource
3. Hook services to register new APIs

Structure overview



The picture above shows the basic structure to extend your app to support Core Mobile API. Take a look at highlighted items and let go to the detail of each section

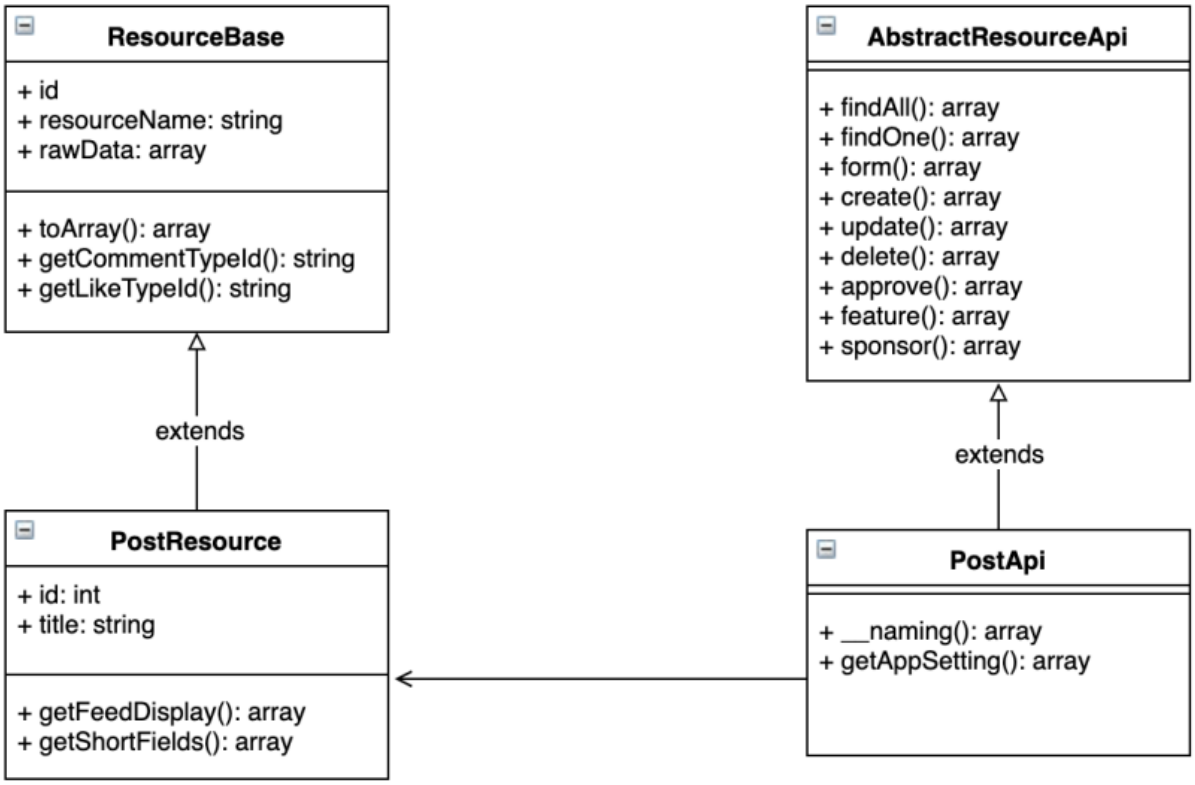
Section	Mission
Api /Resource	This section defines all the resources in your app. Each resource has fields and method to control the application login and API response result. Core Mobile API app provide ResourceBase class for extendable and make it easy to help create standard resource APIs for Native Mobile Application
Api/Form	Create form extend from Mobile API app to help generate form structure that can understand by Mobile Application
Api/Security	The convenient way to control permission to access APIs
Service	Handle your API request and return the response
hooks	Register your APIs or Extent core API

Restful API routing & service handler

Each API routing contains information on how to map HTTP request with a service method handler
API Url pattern: **`url`/restful_api/mobile/`route_path`?access_token=`token`**

- `url` : your PHPFOX website
- `token`: access token generate by AUTH API use to access API
- `route_path` : Define by your new application

We can define new restful API routing in several ways. Because restful API is closely related to a resource, it should be generated from resource class or manual definition in case your app has more APIs to implement.



By extends **ResourceBase** class, **PostResource** able to generate resource API routes and links to **PostApi** service implementation as service handler automatically.
Manual routes can be add via **PostApi:__naming()** method. See example:
PostApi.php

```
<?php
class PostApi extends AbstractResourceApi implements ActivityFeedInterface, MobileAppSettingInterface
{
/.../
        public function __naming()
        {
                return [
                /* API route `/restful_api/mobile/post/search-form` will map with PostApi::searchForm() method
*/
                        'post/search-form'=>[
                                'get'=>'searchForm'
                        ],
                ];
        }
}
```

The final step to help system understands new routes by implement hook "mobile_api_routing_registration.php" (example implement show in the section below).

Following table show detail Resource API mapping were generated from **PostResource** and **PostApi** implementation

Route path pattern	Request Method	Map Method
/post	GET	PostApi::findAll()
/post/:id	GET	PostApi::findOne()
/post/form	GET	PostApi::form()
/post/form/:id	GET	PostApi::form()
/post	POST	PostApi::create()
/post/:id	PUT	PostAPI::update()
/post/:id	DELETE	PostApi::delete()
/post/feature/:id	PUT	PostApi::feature()
/post/approve/:id	PUT	PostApi::approve()
/post/sponsor/:id	PUT	PostApi::sponsor()

Define your resources

Post app has two resources **post** and **post-category,** User can be only able to create post resource
Now let create your post resource. Create new class **PostResource** extend from **ResourceBase.**
The resource's properties are auto map base on same name rule
PostResource.php

```php
<?php

namespace Apps\Posts\Api\Resource;
/* ... */
class PostResource extends ResourceBase
{
        /**
        Define the unique resource name
        */
        const RESOURCE_NAME = "post";
        const TAG_CATEGORY = 'post';
        public $resource_name = self::RESOURCE_NAME;
        public $module_name = 'post';
        /**
        @var string Post's title mapping
        */
        public $title;
        /**
        @var string Post's description mapping
        */
        public $description;
        /**
        @var string Post's parent module id
        */
        public $module_id;
        /**
        @var int Post's parent id
        */
        public $item_id;
        /**
        @var bool Status of the post
        */
        public $is_approved;
        /**
        @var bool sponsor status of the post
        */
        public $is_sponsor;
        /**
        @var bool sponsor status of the post
        */
        public $is_featured;
        public $is_liked;
        public $is_friend;
        public $post_status;
        public $text;
```

Override **ResourceBase::loadMetadataSchema()** method to define database for each field when mapping

```php
<?php

/* A lot of code above */
class PostResource extends ResourceBase
{
/* ... */
        protected function loadMetadataSchema(ResourceMetadata $metadata = null)
        {
                parent::loadMetadataSchema($metadata);
                $this->metadata
                        ->mapField('title', ['type' => ResourceMetadata::STRING])
                        ->mapField('description', ['type' => ResourceMetadata::STRING])
                        ->mapField('item_id', ['type' => ResourceMetadata::INTEGER])
                        ->mapField('module_id', ['type' => ResourceMetadata::STRING])
                        ->mapField('is_approved', ['type' => ResourceMetadata::BOOL])
                        ->mapField('is_sponsor', ['type' => ResourceMetadata::BOOL])
                        ->mapField('is_featured', ['type' => ResourceMetadata::BOOL])
                        ->mapField('is_liked', ['type' => ResourceMetadata::BOOL])
                        ->mapField('is_friend', ['type' => ResourceMetadata::BOOL])
                        ->mapField('post_status', ['type' => ResourceMetadata::INTEGER]);
        }
}
```

If mapping base on the field's name and data type are not enough. We can override the property's value via getting/setting method

```php
<?php

/* A lot of code above */
class PostResource extends ResourceBase
{
/* ... */
/**
Get detail url
@return string
*/
        public function getLink()
        {
                return \Phpfox::permalink('post', $this->id, $this->title);
        }
        public function getImage()
        {
                return Image::createFrom([
                        'file' => $this->rawData['image_path'],
                        'server_id' => $this->rawData['server_id'],
                        'path' => 'post.url_photo',
                        'suffix' => '_1024'
                ]);
        }

        public function getCategories()
        {
                return $this->categories;
        }
        public function getTags()
        {
                return $this->tags;
        }
        public function getText()
        {
                if (empty($this->text) && !empty($this->rawData['text'])) {
                        $this->text = $this->rawData['text'];
                }
                TextFilter::pureHtml($this->text, true);
                return $this->text;
        }
    /* ... */
}
```

Define API service to handle API requests

API Service is similar to PHPFOX App's Service, you can extend from the parent class to minimal and reuse code
Create **PostApi.php** within Service folder
PostApi.php

```
<?php
namespace Apps\Posts\Service;
/* ... */
class PostApi extends AbstractResourceApi implements ActivityFeedInterface, MobileAppSettingInterface
{
        private $postService;
        /**
        @var Process
        */
        private $processService;
        /**
        @var Category
        */
        private $categoryService;
        /**
        @var \User_Service_User
        */
        private $userService;

        public function __construct()
        {
                parent::__construct();
                $this->postService = Phpfox::getService("post");
                $this->categoryService = Phpfox::getService('post.category');
                $this->processService = Phpfox::getService('post.process');
                $this->userService = Phpfox::getService('user');
        }
    /* .. */
}
```

Describe the example above:

- The PostApi class extent AbstractResourceApi to reuse all features
- Implement ActivityFeedInterface to able display the resource to Activity Feed page
- Implement MobileAppSettingInterface to register more screens and actions to Mobile App without change code

Now you need to implement all abstract method from parent class and interfaces.
The initial code of your service would look like below:

PostApi.php

```
<?php

namespace Apps\Posts\Service;
/* ... */
class PostApi extends AbstractResourceApi implements ActivityFeedInterface, MobileAppSettingInterface
{
        /**
         * Get list post
         *
         * @param array $params
         * @return array|mixed
         * @throws ValidationErrorException
         */
        function findAll($params = [])
        {
            /* ... */
                $aItems = $this->browse()->getRows();
                if ($aItems) {
                        $this->processRows($aItems);
                }
                return $this->success($aItems);
        }

    /**
         * @param $params
         * @return array|bool
```

```
     */
    function findOne($params)
    {
            $id = $this->resolver->resolveId($params);
            /* ... */
            return $this->success($resource->loadFeedParam()->toArray());
    }

    public function delete($params)
    {
            $id = $this->resolver->resolveId($params);
            $result = Phpfox::getService('post.process')->delete($id);
            if ($result !== false) {
                    return $this->success([
                            'id' => $id
                    ]);
            }
            return $this->error('Cannot delete post');
    }

/**
     * Get Create/Update document form
     * @param array $params
     * @return mixed
     * @throws \Exception
     */
    public function form($params = [])
    {
            /** @var PostForm $form */
            $form = $this->createForm(PostForm::class, [
                    'title' => 'adding_a_new_post',
                    'method' => 'post',
                    'action' => UrlUtility::makeApiUrl('post')
            ]);
            /* ... */
            return $this->success($form->getFormStructure());
    }
    /**
     * Create a new Post API
     * @param array $params
     * @return array|bool|mixed
     */
    public function create($params = [])
    {
      /* ... */
    }

/**
     * Update a post
     *
     * @param $params
     * @return mixed
     */
    public function update($params)
    {
     /* ... */
    }

    /**
     * @param $id
     * @param bool $returnResource
     * @return array|PostResource
     */
    function loadResourceById($id, $returnResource = false)
    {
            $item = Phpfox::getService("post")->getPost($id);
            if (empty($item['post_id'])) {
                    return null;
            }
            if ($returnResource) {
                    return $this->processOne($item);
```

```
                }
                return $item;
        }
        /**
         * Update multiple document base on document query
         *
         * @param $params
         * @return mixed
         * @throws \Exception
         */
        public function patchUpdate($params)
        {
                /* ... */
        }


        /**
         * Get for display on activity feed
         * @param array $feed
         * @param array $item detail data from database
         * @return array
         */
        public function getFeedDisplay($feed, $item)
        {
                /* ... */
        }

        /**
         * Create custom access control layer
         */
        public function createAccessControl()
        {
                $this->accessControl = new PostAccessControl($this->getSetting(), $this->getUser());
                /* ... */
        }

    /**
         * @param array $params
         * @return mixed
         */
        function searchForm($params = [])
        {
                $this->denyAccessUnlessGranted(PostAccessControl::VIEW);
                /** @var PostSearchForm $form */
                $form = $this->createForm(PostSearchForm::class, [
                        'title' =>'search',
                        'method' => 'GET',
                        'action' => UrlUtility::makeApiUrl('post')
                ]);
                return $this->success($form->getFormStructure());
        }

        public function getRouteMap()
        {
                /* ... */
        }

    /**
         * @param $param
         * @return MobileApp
         */
        public function getAppSetting($param)
        {
                /* ... */
        }
}
```

Register services and resources

After completed create required class and implementation, you need to register your service and resources
Open start.php in your app and add the following row to register service.
start.php

```
<?php

/* ... */
Phpfox::getLib('module')
        ->addAliasNames('post', 'Posts')
        ->addServiceNames([

    // New API service register here
        'mobile.post_api' => Service\PostApi::class,
        'mobile.post_category_api' => Service\PostCategoryApi::class,

    // Other Services of the app
        'post.category' => Service\Category\Category::class,
        'post.category.process' => Service\Category\Process::class,
        'post.api' => Service\Api::class,
        'post' => Service\Posts::class,
        'post.browse' => Service\Browse::class,
        'post.cache.remove' => Service\Cache\Remove::class,
        'post.callback' => Service\Callback::class,
        'post.process' => Service\Process::class,
        'post.permission' => Service\Permission::class,
        ]);
/* ... */
```

Next step, you need to create hook "mobile_api_routing_registration.php" in the hooks folder.
mobile_api_routing_registration.php

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

Extend routes to support more APIs

After complete register API step, your Phpfox site now extend more restful APIs follow naming convention rules as mentions above
If you would like to build more APIs that Resource Naming Convention rules has not supports. In the API service, there a magic function called "__naming()" able to do that
PostApi.php

```php
<?php

class PostApi extends AbstractResourceApi implements ActivityFeedInterface, MobileAppSettingInterface
{
    /**
    This method allow you add custom route for APIs
    Return an array with key is routing rule and mapping condition
    In this case, 'mobile/post/search-form' map to `searchForm` method
    */
    public function __naming()
    {
            return [
                    'post/search-form'=> [
                    'get'=>'searchForm'
                    ]
            ];
    }

    /**
    @param array $params
    @return mixed
    */
    function searchForm($params = [])
    {
            $this->denyAccessUnlessGranted(PostAccessControl::VIEW);
            /** @var PostSearchForm $form */
            $form = $this->createForm(PostSearchForm::class, [
                    'title' =>'search',
                    'method' => 'GET',
                    'action' => UrlUtility::makeApiUrl('post')
                    ]);
            return $this->success($form->getFormStructure());
    }
}
```

You can download the full sample code of Post API service.

Testing Your APIs

Postman is one of a good application for testing APIs.
Here are the API patterns to access Posts app

Pattern	description	ex. response

`GET {{url}}/restful_api/mobile/post?access_token={{token}}`	Get List of Posts	{
		"status": "success",
		"data": [
		{
		"resource_name": "post",
		"title": "Title of the post",
		"description": "Description of the post",
		"module_id": "post",
		"item_id": 0,
		"is_approved": true,
		"is_sponsor": false,
		"is_featured": false,
		"is_liked": null,
		"is_friend": null,
		"post_status": 1,
		"image": null,
		"statistic": {
		"total_like": 0,
		"total_comment": 0,
		"total_view": 1,
		"total_attachment": 0
		},
		"privacy": 0,
		"user": {
		"full_name": "Cute User",
		"avatar": null,
		"id": 1
		},
		"categories": [
		{
		"id": 6,
		"name": "Recreation"
		}
		],
		"tags": [],
		"attachments": [],
		"id": 1,
		"creation_date": "2018-11-30T04:44:39+00:00",
		"modification_date": null,
		"link": "http://localhost:7788/post/1/love/",
		"extra": {
		"can_view": true,
		"can_like": true,
		"can_share": true,
		"can_delete": true,
		"can_report": false,
		"can_add": true,
		"can_edit": true,
		"can_comment": true,
		"can_publish": false,
		"can_feature": true,
		"can_approve": true,
		"can_sponsor": true,
		"can_sponsor_in_feed": false,
		"can_purchase_sponsor": true
		},
		"self": null,
		"links": null
		}
		]
		}

	Get Post Detail	

```
GET {{url}}/restful_api/mobile/post/1?
access_token={{token}}
```


```
{
    "status": "success",
    "data": {
        "resource_name": "post",
        "title": "Title of the post",
        "description": "Description of the
post",
        "module_id": "post",
        "item_id": 0,
        "is_approved": true,
        "is_sponsor": false,
        "is_featured": false,
        "is_liked": false,
        "is_friend": false,
        "post_status": 1,
        "text": "xxxx",
        "image": null,
        "statistic": {
            "total_like": 0,
            "total_comment": 0,
            "total_view": 1,
            "total_attachment": 0
        },
        "privacy": 0,
        "user": {
            "full_name": "Cute User",
            "avatar": null,
            "id": 1
        },
        "categories": [
            {
                "id": 6,
                "name": "Recreation"
            }
        ],
        "tags": [],
        "attachments": [],
        "id": 1,
        "creation_date": "2018-11-30T04:44:
39+00:00",
        "modification_date": null,
        "link": "http://localhost:7788/post/1
/love/",
        "extra": {
            "can_view": true,
            "can_like": true,
            "can_share": true,
            "can_delete": true,
            "can_report": false,
            "can_add": true,
            "can_edit": true,
            "can_comment": true,
            "can_publish": false,
            "can_feature": true,
            "can_approve": true,
            "can_sponsor": true,
            "can_sponsor_in_feed": false,
            "can_purchase_sponsor": true
        },
        "self": null,
        "links": {
            "likes": {
                "ref": "mobile/like?
item_type=post&item_id=1"
            },
            "comments": {
                "ref": "mobile/comment?
item_type=post&item_id=1"
            }
        },
        "feed_param": {
```

		"item_id": 1, "comment_type_id": "post", "total_comment": 0, "like_type_id": "post", "total_like": 0, "feed_title": "Love", "feed_link": "http://localhost:7788/post/1/love/", "feed_is_liked": false, "feed_is_friend": false, "report_module": "post" } } }

`GET {{url}}/restful_api/mobile/post/form/?access_token={{token}}`
 	
`# Or get Edit form response`
`GET {{url}}/restful_api/mobile/post/form/1?access_token={{token}}`
	Get create post form	{
		"status": "success",
		"data": {
		"title": "Add a new post",
		"description": "",
		"action": "mobile/post",
		"method": "post",
		"sections": {
		"basic": {
		"label": "Basic Info",
		"fields": {
		"title": {
		"name": "title",
		"component_name": "Text",
		"required": true,
		"value": "",
		"returnKeyType": "next",
		"label": "Title",
		"placeholder": "Fill title for post"
		},
		"text": {
		"name": "text",
		"component_name": "TextArea",
		"required": true,
		"value": "",
		"returnKeyType": "default",
		"label": "Post",
		"placeholder": "Add content to post"
		},
		"attachment": {
		"name": "attachment",
		"component_name": "Attachment",
		"label": "attachment",
		"item_type": "post",
		"item_id": null,
		"current_attachments": null,
		"upload_endpoint": "mobile/attachment"
		}
		}
		},
		"additional_info": {
		"label": "Additional Info",
		"fields": {

```
                    "categories": {
                        "name": "categories",
                        "component_name":
"Choice",
                        "multiple": true,
                        "value": [],
                        "label": "Categories",
                        "value_type": "array",
                        "options": [
                            {
                                "value": 1,
                                "label":
"Business"
                            },
                            {
                                "value": 2,
                                "label":
"Education"
                            },
                            {
                                "value": 3,
                                "label":
"Entertainment"
                            }
                        ],
                        "suboptions": []
                    },
                    "tags": {
                        "name": "tags",
                        "component_name":
"Tags",
                        "value": "",
                        "returnKeyType":
"next",
                        "label": "Topics",
                        "description":
"Separate multiple topics with commas."
                    },
                    "file": {
                        "name": "file",
                        "component_name":
"File",
                        "label": "Photo",
                        "file_type": "photo",
                        "item_type": "post",
                        "preview_url": null,
                        "upload_endpoint":
"mobile/file"
                    }
                }
            },
            "settings": {
                "label": "Settings",
                "fields": {
                    "privacy": {
                        "name": "privacy",
                        "component_name":
"Privacy",
                        "value": 0,
                        "returnKeyType":
"next",
                        "options": [
                            {
                                "label":
"Everyone",
                                "value": 0
                            },
                            {
                                "label":
"Friends",
                                "value": 1
```

<table>
<tr>
<td></td>
<td></td>
<td>
<pre>
                                },
                                {
                                    "label":
"Friends of Friends",
                                    "value": 2
                                },
                                {
                                    "label": "Only
Me",
                                    "value": 3
                                },
                                {
                                    "label":
"Custom",
                                    "value": 4
                                }
                            ],
                            "label": "Privacy",
                            "multiple": false,
                            "description":
"Control who can see this post"
                        }
                    }
                }
            },
            "fields": {
                "module_id": {
                    "name": "module_id",
                    "component_name": "Hidden",
                    "value": "post"
                },
                "item_id": {
                    "name": "item_id",
                    "component_name": "Hidden"
                },
                "draft": {
                    "name": "draft",
                    "component_name": "Checkbox",
                    "value": 0,
                    "label": "Save as Draft"
                },
                "submit": {
                    "name": "submit",
                    "component_name": "Submit",
                    "label": "Publish",
                    "value": 1
                }
            }
        }
}
</pre>
</td>
</tr>
<tr>
<td>
<pre>
DELETE {{url}}/restful_api/mobile/post/
{{post_id}}?access_token={{token}}
</pre>
</td>
<td>Delete a Post</td>
<td></td>
</tr>
</table>