eCommerce

Components

Files

UI

Performance

Framework

PHP

- Learn PHP
- PHP Introduction
- PHP Basics
- Contact Form
- Comments System
- Types, Variables & Operators
- PHP Strings
- PHP Arrays
- PHP Functions
- PHP OOPS
- PHP Forms
- Advanced
- PHP AJAX
- RESTful API
  - **PHP RESTful Web Service API - Part 1 - Introduction with Step-by-step Example**
  - PHP MySQL REST API for Android
  - REST API CRUD using PHP
  - REST API Search Implementation in PHP
- PHP Databases
- PHP Sessions and Cookies
- Error and Exception Handling
- File Upload
- Files and Directories

PHP RESTful Web Service API – Part 1 – Introduction with Step-by-step Example

by Vincy. Last modified on May 27th, 2021.

This is part 1 of a three part series to help you learn RESTful web services us PHP. These tutorials will be comprehensive, by following it through you can b your own web services easily and consume external services.

In this tutorial, we will see how to create PHP RESTful web service without us any framework. Most of the times I do prefer to write custom code without depending on frameworks since this approach has lot of advantages. Mainly this will take you deeper in learning the concepts and you can keep things sle and effective.

REST or **Representational State Transfer** is one of the popular architectural s used to develop web services. This style of architecture contains constraints rules to design web services which can be accessed from external apps or we applications.

The objective of this PHP RESTful web service example

The objective is to build a RESTful web service in PHP to provide resource da based on the request with the network call by the external clients. Also, the following list of steps are implemented while customizing this example witho depending on any framework.

- Create request URI with patterns that follow REST principles.
- Make the RESTful service to be capable of responding to the requests i JSON, XML, HTML formats.
- Demonstrate the use of HTTP Status code based on different scenarios
- Demonstrate the use of Request Headers.
- Test the RESTful web service using a REST client.

How it is made?

An array of mobile names are the resource data that will be targeted by the R clients. I have this resource in a domain class of this PHP RESTful example.

For accessing these data via this web service, the client will send the request setting URI, parameters with the selected method, and more information.

The resource handlers of the web service will prepare the response in JSON, X or HTML format based on the request. Then, the response will be sent to the client.

On the Internet, I have seen web services tutorials and most of the times they turn out to be error-prone or incomplete. I tested those RESTful services using REST client and mostly they fail.

view demo

What is inside?

1. What is RESTful?
2. RESTful web services vs RPC web services
3. RESTful web services API architecture
4. Uses of RESTful API
5. PHP RESTful web service example
6. File structure of RESTful example service
7. RESTful services URI mapping
8. UML sequence diagram for the example RESTful service
9. RESTful web service controller
10. A simple RESTful base class
11. RESTful web service handler
12. RESTful web service client
13. RESTful Web Service Output

What is RESTful?

REST stands for Representational State Transfer and it is an architectural sty that enables communication between systems. The term REST was first coin by Roy T. Fielding in his PhD. dissertation.

The concept of REST is defined by certain rules, constraints or principles. The system, application, services or whatever satisfies these REST principles are

called RESTful.

Web services that follow the RESTful principles are RESTful services. The UR used to access RESTful services to get the resources.

In the RESTful glossary, the resources are nothing but the data and functions So eventually we will call the web services via URI to access functions and thereby get the resource data.

REST Constraints

The following constraints define the RESTfulness of an application or service

- Client-Server architecture
- Statelessness
- Uniform interface
- Layered system
- Cacheability
- Code on Demand

RESTful web services vs RPC web services

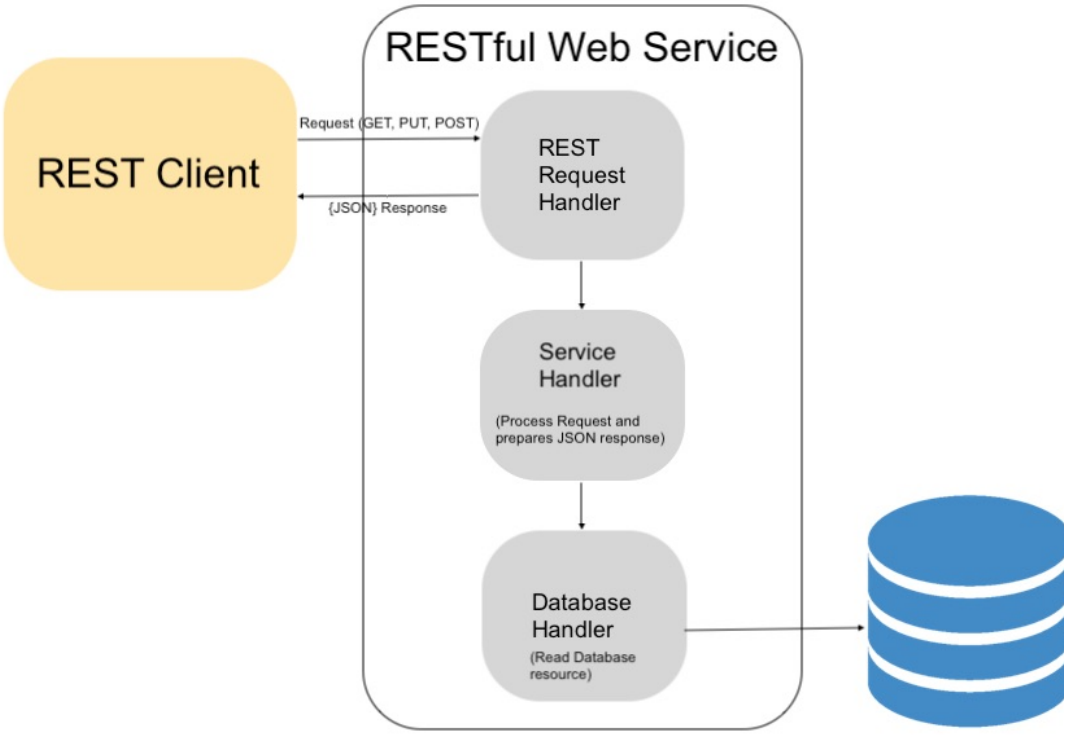
The following table shows the comparison of RESTful and RPC style web services. This comparison is made by factors like service request URI, reques methods, data transmission, service handlers and more.

	RESTful-Style	**RPC-Style**
Request URI	The request URI will differ based on the resource.	Same URI for all resources.
Request methods	The service request parameters can be sent via GET, PUT, POST request methods.	Supports only the POST method.
Service methods or handlers	Same method for all resources.	Methods and params are posted o request.
Target on	The service request using this style targets resources.	The target is methods.
Response data transmission	Over HTTP	wrapped with the requested metho and params.

RESTful web services API architecture

The following diagram shows a RESTful web service architecture. In this diagram, the request-response flow among the client-server is represented.

In this diagram, the database is shown as a resource. Based on the web servi
the resource can be XML feed, JSON data extracted from the file system or a



Uses of RESTful API

RESTful API provides services to access resources from external applications
REST clients. Some of the predominant uses of the RESTful API is listed belo

- As an interface with multi-platform support which is used to access resources from outside application coded in various programming languages like PHP, JAVA, Android and more.
- REST is the simple architectural style for transmitting data over HTTP.
- The REST API is the most suitable resource provider for an AJAX-based application interface which requires data to update UI without page relo
- By meeting more the REST constraints, the web applications or services can support a wide range of clients.

PHP RESTful web service example

In the PHP RESTful web service example, the following domain class contain
the resource data array and service handlers. These handlers are called base
the request sent by the REST client or external apps.

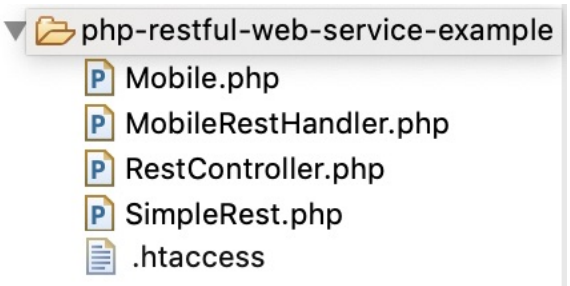
In the next section, we can see all the file structure and the purpose of each fi this example.

```
<?php
/*
A domain Class to demonstrate RESTful web services
*/
Class Mobile {

    private $mobiles = array(
        1 => 'Apple iPhone 6S',
        2 => 'Samsung Galaxy S6',
        3 => 'Apple iPhone 6S Plus',
        4 => 'LG G4',
        5 => 'Samsung Galaxy S6 edge',
        6 => 'OnePlus 2');

    /*
        you should hookup the DAO here
    */
    public function getAllMobile(){
        return $this->mobiles;
    }

    public function getMobile($id){

        $mobile = array($id => ($this->mobiles[$id]) ? $this->mobiles[$id] :
        return $mobile;
    }
```

File structure of RESTful example service

Below file structure shows the simplicity of creating a RESTful web service example.

As discussed above the Mobile.php is the domain class which is having reso array and handlers to get the resource.



The .htaccess file is used for mapping the request URI to the REST service endpoint.

In the following sections, we will see how the URI is mapped, how the service handler is invoked to get resource data from the domain. – URI RFC 3986

RESTful services URI mapping

Every resource is identified via a URI (Uniform Resource Identifier).

> A Uniform Resource Identifier (URI) is a compact sequence of characters that identifies an abstract or physical resource.

RestController.php shown in the above file structure is the PHP endpoint to w the request is to be forwarded.

In this example, I provide two URIs for accessing this web service from extern applications or REST client. One URI will be used to get the complete array of mobile names in a JSON format and the other is to get a particular mobile na based on the ident passed via the request URI.

URI	Method	Type	Description
http://localhost/restex ample/mobile/list/	GET	JSON	To get the list of m names in an JSON array.
http://localhost/restex ample/mobile/list/{id}/	GET	JSON	To get single mobil data array by ident passed via URL.

The following URIs are mapped to the real file via the .htaccess file.

URI to get the list of all mobiles:

```
http://localhost/restexample/mobile/list/
```

URI to get a particular mobile's detail using its id:

In the below URI the number '2' is the id of a mobile. The resource domain cla can get the particular data with the reference of this id parameter.

```
http://localhost/restexample/mobile/list/2/
```

Below code snippet shows the complete rules and URL mappings created for PHP RESTful web service example in its `.htaccess` file.

```
# Turn rewrite engine on
Options +FollowSymlinks
RewriteEngine on

# map neat URL to internal URL
RewriteRule ^mobile/list/$   RestController.php?view=all [nc,qsa]
RewriteRule ^mobile/list/([0-9]+)/$   RestController.php?view=single&id=$1 [
```

RESTful web service controller

In the `.htaccess` file, we are forwarding all the request to the RestController.ph file.

While forwarding the request the parameters are sent to execute a required pa of the REST controller. This parameter is the key named as 'view'.

The value of the key parameter can be either "all" or "single" based on the request URI.

Following is the RestController.php file that receives the request and gets the view parameter. Based on this parameter value, the appropriate controll case be executed.

On the controller cases, the request is dispatched to respective methods creat in the REST handler class.

```
<?php
require_once("MobileRestHandler.php");

$view = "";
if(isset($_GET["view"]))
    $view = $_GET["view"];
/*
controls the RESTful services
URL mapping
*/
switch($view){

    case "all":
        // to handle REST Url /mobile/list/
        $mobileRestHandler = new MobileRestHandler();
        $mobileRestHandler->getAllMobiles();
        break;

    case "single":
        // to handle REST Url /mobile/show/<id>/
        $mobileRestHandler = new MobileRestHandler();
        $mobileRestHandler->getMobile($_GET["id"]);
        break;

    case "" :
        //404 - not found;
        break;
```

A simple RESTful base class

Following class has a couple of methods that can be commonly used for the RESTful service handlers.

The getHttpStatusMessage() method is used to get the HTTP status messag construct the response. It contains the HTTP status code and message mapp array.

By receiving the status code, it returns the appropriate header response mess If the invalid status code is passed to this function or no such code is found i the mapping array, then the “Invalid Server Error” will be returned in the respo

These methods can be commonly used in the base class of simple PHP RES web services.

```
<?php
/*
A simple RESTful webservices base class
Use this as a template and build upon it
*/
class SimpleRest {

    private $httpVersion = "HTTP/1.1";

    public function setHttpHeaders($contentType, $statusCode){

        $statusMessage = $this -> getHttpStatusMessage($statusCode);

        header($this->httpVersion. " ". $statusCode ." ". $statusMessage);
        header("Content-Type:". $contentType);
    }

    public function getHttpStatusMessage($statusCode){
        $httpStatus = array(
            100 => 'Continue',
            101 => 'Switching Protocols',
            200 => 'OK',
            201 => 'Created',
            202 => 'Accepted',
            203 => 'Non-Authoritative Information',
            204 => 'No Content'
```

RESTful web service handler

This is the service class of this PHP example that handles the REST request dispatched from the controller.

First, we have to decide about the response format in which the resource data has to be prepared. It is based on the request header parameters.

In the request header, the “Accept” parameter will have the specification abou the response content format or type.

The protocol here is, when the request is sent, it should set the Request heade parameter “Accept” and send it. The values can be like “application/json” or

“application/xml” or “text/html”.

Based on these values the response data will be ready by invoking appropriat methods encodeJson(), encodeXML(), encodeHTML() shown below.

Then, the status code has to be returned to the client with the response data. success, the status code will be 200.

Similarly, there are different status codes available and they should be used accordingly to set response header as we discussed in the above section.

```
<?php
require_once("SimpleRest.php");
require_once("Mobile.php");

class MobileRestHandler extends SimpleRest {

    function getAllMobiles() {

        $mobile = new Mobile();
        $rawData = $mobile->getAllMobile();

        if(empty($rawData)) {
            $statusCode = 404;
            $rawData = array('error' => 'No mobiles found!');
        } else {
            $statusCode = 200;
        }

        $requestContentType = $_SERVER['HTTP_ACCEPT'];
        $this ->setHttpHeaders($requestContentType, $statusCode);

        if(strpos($requestContentType,'application/json') !== false){
            $response = $this->encodeJson($rawData);
            echo $response;
        } else if(strpos($requestContentType,'text/html') !== false){
            $response = $this->encodeHtml($rawData);
```

RESTful web service client

There are various stand-alone REST clients available in the market. These cli interfaces are used to test a RESTful web service.

The Advanced Rest Client extension which can be added to the Chrome insta in your machine.

We can also write our own custom client to test a RESTful web service.

I used this Google Chrome extension REST client for testing this PHP RESTfu web service example.

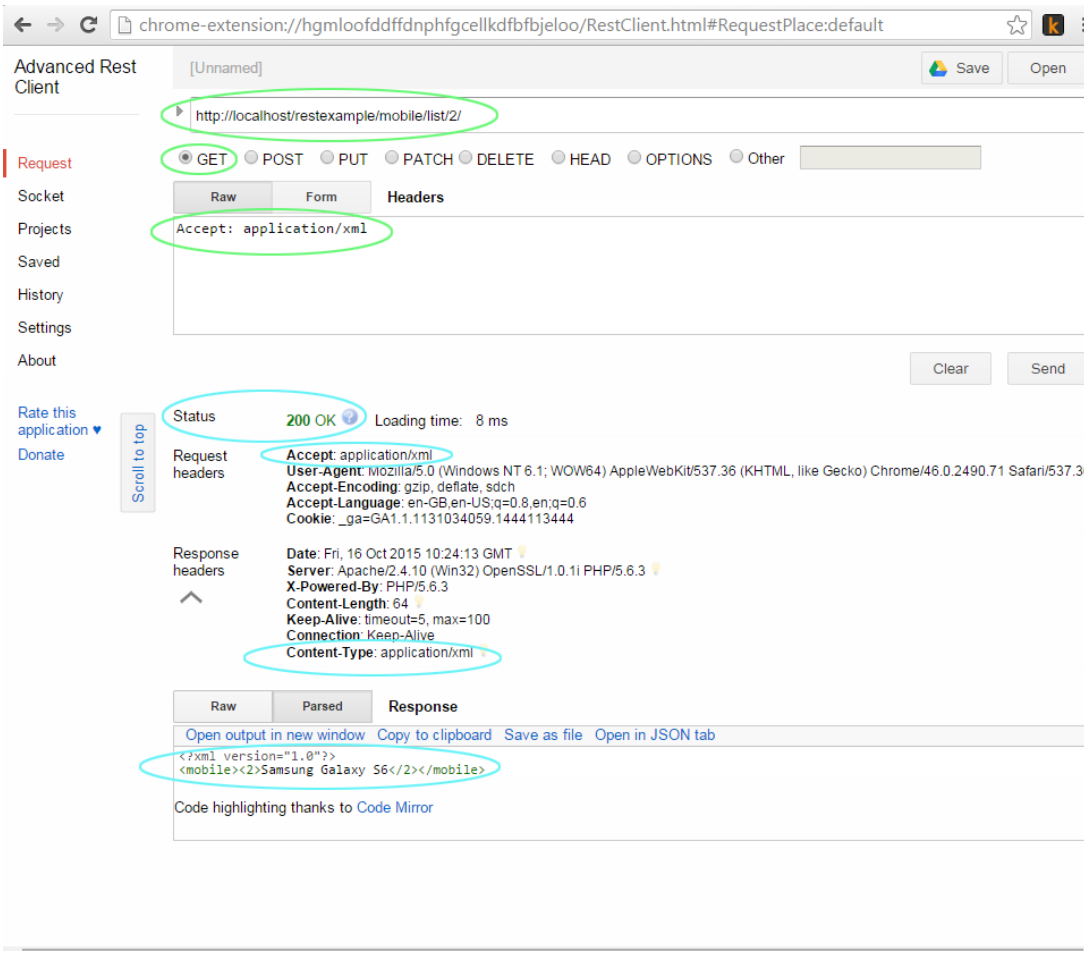
PHP RESTful web service output

In the below screenshot, it shows how to call RESTful web service. In this screenshot, the circled sections highlight the request URI, selected request method, Header's Accept param, and more details.

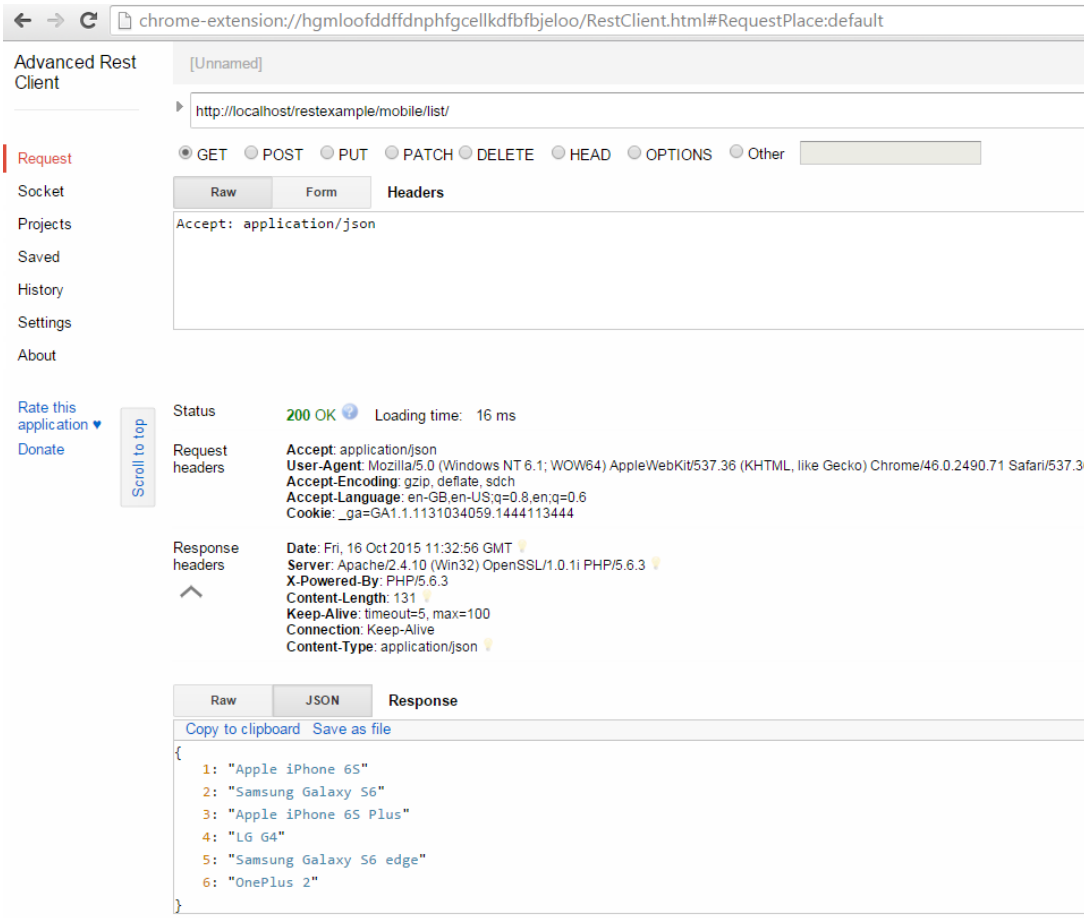
By clicking the send button, the response will be returned from the PHP REST web service.

XML Response

I set the **application/xml** as the response type. So the resultant resource data prepared in the requested format as shown in the response section of the bel screenshot.



PHP RESTful web service JSON response



Conclusion

In this three part tutorial series on RESTful webservices using PHP, you will le the RESTful implementation in detail using these comprehensive material. Th first part has given you a complete introduction to the concepts with step by s example.

With the knowledge that you have acquired from this tutorial, about the rules principles of RESTfulness, you can build a RESTful API easily. Though there a frameworks for developing RESTful API, it can be done by using plain core PH which will be effective and provide good performance.

In the coming part, you will be seeing about all aspects of developing a CRUD RESTful web services API using PHP for an entity.

view demo download

Comments to “PHP RESTful Web Service API – Pa 1 – Introduction with Step-by-step Example”

Emanuele Magni
March 16, 2019 at 3:15 pm

Very good example.

Reply

Vincy
March 17, 2019 at 1:58 pm

Thank you Emanuele.

Reply

Jonathan Cruz Tejada
May 22, 2019 at 3:21 am

iTS BEAUTIFUL EXAMPLE, THANKS

Reply

Vincy
May 23, 2019 at 10:26 am

Welcome Jonathan.

mahmoud
April 15, 2022 at 11:22 pm

Hi
it was good
thank you

Reply

Vincy
May 6, 2022 at 11:09 am

Welcome Mahmoud.

Claude
March 29, 2019 at 8:01 pm

Thank you but where are the part 2 and 3 ?

Reply

Vincy
March 30, 2019 at 12:35 pm

Welcome Claude. It will be updated in a month.

Reply

Lau
May 8, 2019 at 2:12 pm

Ohh I hope it will come soon :) Thanks for the first part!

Rajendra
April 30, 2019 at 8:19 pm

thank you. ms. vincy your blog is very nice and with very simple example whi help me in most of the project. after learning basic from here I am easily able apply on my required logic… thank you once again.

Reply

Vincy
May 1, 2019 at 2:47 pm

Welcome, Rajendra for the nice words. It made my day! Thank you.

Reply

Anarbona
May 1, 2019 at 10:28 pm

Hello Vincy, thank you for your tutorial, I think it's a fantastic material. I look forward to seeing the other two parts.
A greeting from Spain ;)

Reply

Vincy
May 2, 2019 at 2:47 pm

Hey Anarbona, thank you so much. I will get the other two parts soor

Reply

ANUJ KUMAR
May 8, 2019 at 11:06 am

Very easy and understandable example .. Thanks a lot

Reply

Vincy
May 9, 2019 at 2:46 pm

Welcome, Anuj.

Reply

Huseyin
May 20, 2019 at 3:42 pm

Hi Vincy
Thank you for your awesome tutorial.. It is so helpful..

I changed some lines;

MobileRestHandler.php:30
added else{}

SimpleRest.php:61
return ($httpStatus[$statusCode]) ? $httpStatus[$statusCode] : $httpStatus[5

Reply

Vincy
May 21, 2019 at 10:28 am

Welcome, Huseyin. Thanks for sharing the updated code snippet.

Thank you so much for adding value. When you do such things, the Internet becomes a beautiful place. Thank you for your time.

Reply

Rakesh Yadav
June 5, 2019 at 3:42 pm

Example was good. But still where are the next tutorial. I have been eagerly waiting !!

Reply

Vincy
June 6, 2019 at 2:59 pm

Thanks Rakesh. Coming soon!

Reply

Nadeem Inamdar
July 15, 2019 at 5:18 pm

Very well explained. But you missed out in the SQL for creating the tbl_mobile table. Please include it for the readers and learners.

Reply

Vincy
July 18, 2019 at 11:08 am

Thank you Nadeem, sure I will update the article with that.

Reply

Heather
August 17, 2019 at 9:16 am

This is a helpful tutorial. Thank-you for creating and sharing it.

A slight modification to the rewrite rules in the .htaccess file will allow the trai forward slash to be optional when the user enters a URL in the browser. Wher first tested out this example I put http:/localhost/restexample/mobile/list into the browser's address bar. This returned a 404 error. Initially I thought my Apa configuration was not set up correctly to allow overrides by .htaccess but tha was not the case. The problem was that I hadn't put a trailing forward slash a the end of the URL and the original rewrite rules were written such that the trailing forward slash was mandatory.

Here is how I modified the rewrite rules to make the trailing forward slash optional:
RewriteRule ^mobile/list/?$ RestController.php?view=all [nc,qsa]
RewriteRule ^mobile/show/([0-9]+)/?$ RestController.php?view=single&id=$1 [nc,qsa]

Reply

Vincy
August 20, 2019 at 2:45 pm

Hi Heather,

Thank you for the updated code. Appreciate it. Thank you so much for adding value to the article and discussion.

Reply

Demiurge
July 10, 2020 at 11:42 pm

Hi Heather,

thank you for your comment. I'm searching several hours for the proble

greets
Demiurge

Reply

Jignesh
October 31, 2019 at 1:26 pm

Described very well, Never found article about REST before.

Reply

Vincy
May 27, 2021 at 4:48 pm

Thank you Jignesh.

Reply

Abdul Sadeeq
November 2, 2019 at 10:20 am

Wow! Excellent work Ms. Vincy… Since the time i came across your succinct tutorial, i completely feel at rest! I am following your great work here with grea desire and i found it very helpful. Your assistance is greatly appreciated and v commendable!!! Keep it up

Reply

Vincy
May 27, 2021 at 4:47 pm

Thank you Abdul.

Reply

Kapuriya Dixit
November 3, 2019 at 9:03 pm

good

Reply

Vincy
May 27, 2021 at 4:47 pm

Thank you Dixit.

Reply

Konstantinos
November 26, 2019 at 2:42 am

Very good approach!!!
It would be very useful if an example with POST could be given.

Great job.

Reply

Vincy
November 27, 2019 at 4:10 pm

Thank you Konstantinos. Sure I will add it soon.

Reply

Edgar
November 28, 2019 at 11:06 am

Thank you for sharing your knowledge.

Greetings from Guatemala!

Reply

Vincy
May 27, 2021 at 4:47 pm

Welcome Edgar.

Reply

Milan
January 11, 2020 at 1:33 pm

Really useful example – has really help speed up my learning
Thank you

Reply

Vincy
May 27, 2021 at 4:46 pm

Welcome Milan.

Reply

Juan
February 14, 2020 at 6:29 pm

Hi Vinci, thank you very much for your example. It’s a excellent start for my project :)

Reply

Vincy
May 27, 2021 at 4:46 pm

Welcome Juan.

Reply

Roy
March 26, 2020 at 1:52 pm

Hey, I've just got back into development after a long break, some of the conce I'm finding have a very steep learning curve.

You have at least made this one simple for me.

I'm guessing you have finished the parts of the tutorial, can you email me sor links

Reply

Vincy
May 27, 2021 at 4:46 pm

Thank you Roy. I will post them here.

Reply

Qasem
April 28, 2020 at 2:13 am

Thank you Vincy. It was so helpful.

Reply

Vincy
May 27, 2021 at 4:45 pm

Welcome Qasem.

Reply

Amnat Thapchula
April 28, 2020 at 4:34 pm

Very good tutorial. Thank you.

Reply

Vincy
May 27, 2021 at 4:45 pm

Thank you Amnat.

Reply

Bhaskar
May 6, 2020 at 1:51 pm

Awesome article.. just copy pasted its working great. Thanks.

Reply

Vincy
May 27, 2021 at 4:45 pm

Welcome Bhaskar.

Reply

Amos
May 21, 2020 at 2:04 am

Such a blessing from tutorial from you… Live long helping amateurs

Reply

Vincy
May 27, 2021 at 4:44 pm

Thank you Amos.

Reply

Pankaj
June 11, 2020 at 3:44 pm

Just amazing. I was building similar kinds of stuff, and I found the solution i
.htaccess file
Thanks a lot :)

Reply

Vincy
May 27, 2021 at 4:44 pm

Welcome Pankaj.

Reply

Muhammad Ali
July 14, 2020 at 4:07 pm

Perfect you saved a lot of my time. i appreciate your efforts. Image passing v rest post request will be more help…

Reply

Vincy
May 27, 2021 at 4:44 pm

Welcome Muhammad.

Sure I will write a separate article with image passing in REST.

Reply

UA
September 19, 2020 at 5:26 pm

Thank you Vincy.

Reply

Vincy
September 19, 2020 at 6:19 pm

Welcome UA :-)

Reply

Sowjanya

October 13, 2020 at 5:39 pm

Hello Vince, It's a awesome tutorial, explained very clearly and easy to understand.
I wanted to implement it for external application, but how to overcome cross domain access authentication and issues?

Reply

Vincy
May 27, 2021 at 4:43 pm

Thank you Sowjanya.

Cross domain authentication is a topic by itself. I will try to write an article on it soon.

Reply

Ken
December 2, 2020 at 4:59 pm

I found this really useful and built a working POC for a project.

Reply

Vincy
May 27, 2021 at 4:42 pm

Thank you Ken.

Reply

Joe
February 1, 2021 at 7:11 am

You give code but you don't give the name of the files. Please add the name c the files for each code piece.

Reply

Vincy
May 27, 2021 at 4:41 pm

Sure Joe noted.

The complete code is available in the download as a zip. I would recommend you to use that.

Reply

Matt
April 4, 2021 at 5:07 pm

Hi, Cant get a valid response. Header response returns as undefined.

Reply

Vincy
May 27, 2021 at 4:40 pm

Hi Matt,

Can you post the complete trace?

Reply

Jim
May 5, 2021 at 2:31 pm

Great article but a question:

Under the sample code for “A simple RESTful base class”, should:

return ($httpStatus[$statusCode]) ? $httpStatus[$statusCode] : $status[500];

be:

return ($httpStatus[$statusCode]) ? $httpStatus[$statusCode] : $httpStatus[5

Thanks, just what I was looking for!

Reply

Vincy
May 27, 2021 at 4:40 pm

Thank you Jim. Now I have updated the code.

Reply

Vince
April 1, 2022 at 4:52 am

Hi Vincy. It seems everyone who posts an article about creating REST Web Services always employs other technology to get the project started. Bootstra Eclipse, Java… It really takes away from understanding the underlying REST r and bolts.

I love your article because it explains the basis of it all from the ground up. N special glue. Just the developer and a text editor. Wonderful!!

It's now 2022 and I don't see links to the subsequent articles. Are they availab somewhere?

Reply

Vincy
May 6, 2022 at 11:14 am

Thank you Vince. Yet to post the other parts :-(

Reply

Leave a Reply

Comment

Name *

Email *

Post Comment

Popular Articles

- ★ REST API Search Implementation in PHP
- ★ PHP MySQL REST API for Android
- ★ REST API CRUD using PHP

Search articles

↑ Back to