**Batch: B1 Roll No.: 1511070**

**Experiment / assignment / tutorial No. 2**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| Title: Demonstrate the use of REST full web services. |

**AIM:** To learn the development of REST full web services.

**Problem Definition:**

Development of RESTful web services for a website.

**Resources used:** Web Server, PHPMyAdmin XAMPP

https://www.webslesson.info/2018/05/how-to-make-simple-crud-rest-api-in-php-with-mysql.html

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO 2:** Describe the role of RESTful web services in web application development. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Web Services Essentials, Ethan, O’ Reilly Publication

**Pre Lab/ Prior Concepts:**

A web service is any service that is available over the Internet, uses a standardized XML messaging system, and is not tied to any one operating system or programming language. There are several alternatives for XML messaging. For example, you could use XML Remote Procedure Calls (XML-RPC) or SOAP, both of which are described later in this chapter. Alternatively, you could just use HTTP GET/POST and pass arbitrary XML documents. Any of these options can work.

Although they are not required, a web service may also have two additional (and desirable) properties:

* A web service should be self-describing. If you publish a new web service, you should also publish a public interface to the service. At a minimum, your service should include human-readable documentation so that other developers can more easily integrate your service. If you have created a SOAP service, you should also ideally include a public interface written in a common XML grammar. The XML grammar can be used to identify all public methods, method arguments, and return values.
* A web service should be discoverable. If you create a web service, there should be a relatively simple mechanism for you to publish this fact. Likewise, there should be some simple mechanism whereby interested parties can find the service and locate its public interface. The exact mechanism could be via a completely decentralized system or a more logically centralized registry system.

To summarize, a complete web service is, therefore, any service that: ·

* Is available over the Internet or private (intranet) networks ·
* Uses a standardized XML messaging system
* Is not tied to any one operating system or programming language
* Is self-describing via a common XML grammar
* Is discoverable via a simple find mechanism

**RESTful** Web Services are basically REST Architecture based Web Services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web-based applications.

REST stands for Representational State Transfer. REST is web standards based architecture and uses HTTP Protocol. It revolves around resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods. REST was first introduced by Roy Fielding in 2000.

In REST architecture, a REST Server simply provides access to resources and REST client accesses and modifies the resources. Here each resource is identified by URIs/ global IDs. REST uses various representation to represent a resource like text, JSON, XML. JSON is the most popular one.

HTTP methods

Following four HTTP methods are commonly used in REST based architecture.

* **GET** − Provides a read only access to a resource.
* **POST** − Used to create a new resource.
* **DELETE** − Used to remove a resource.
* **PUT** − Used to update a existing resource or create a new resource.

**Methodology:**

Following are the file structure of PHP REST API for CRUD operation.

* api //In this folder we have store PHP API File
* Api.php
* //This is API class for Insert Update Delete
* test\_api.php
* //This file will handle API request.
* work //In this folder we will make file for send API request
* index.php
* fetch.php
* action.php

**Implementation Details: Code attached separately**

**Steps for execution of the code**

1. Login to the localhost, via phpmyadmin.

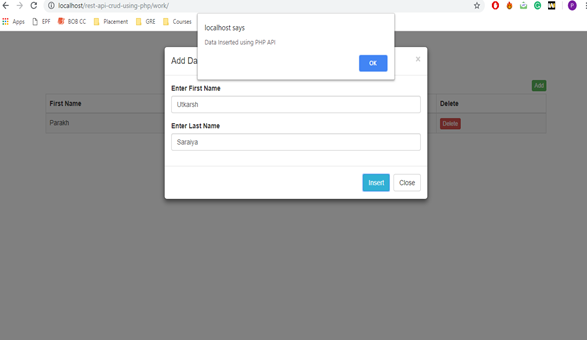
2. Select the file you want to run by providing the path in the url.

3. The selected files will run. Inter-linking between pages has been provided via various means such as nav bars, buttons and address links.

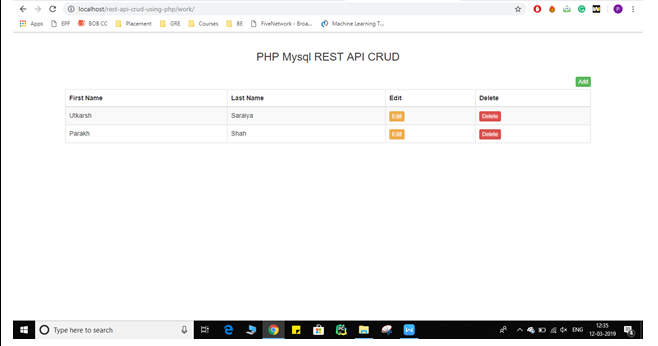
4. Perform the intended tasks as mentioned in the problem definition.

**Output:**

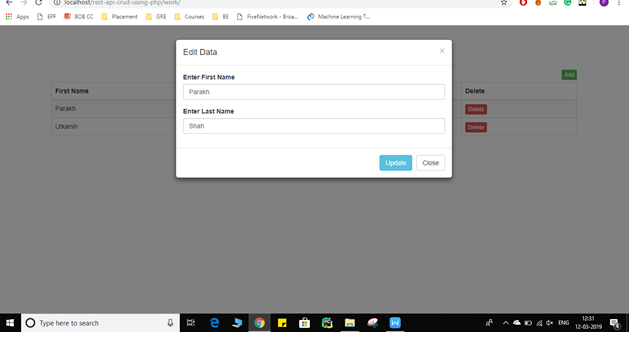
**Create:**



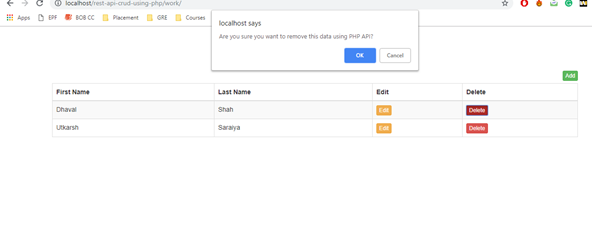
**Read:**



**Update:**



**Delete:**



**Conclusion:** Thus, we have demonstrated the CRUD operations in a PHP environment successfully

**Postlab questions:**

1. **What do you know about RESTful Web Services?**

REST is any interface between systems using HTTP to obtain data and generate operations on those data in all possible formats, such as XML and JSON. This is an increasingly popular alternative to other standard data exchange protocols such as SOAP (Simple Object Access Protocol), which have a high capacity but are also very complex. Sometimes it's preferable to use **a simpler data-processing solution such as REST**

### REST's features

* **Stateless client/server protocol**: each HTTP contains all the necessary information to run it, which means that neither the client nor the server need to remember any previous state to satisfy it. Be that as it may, some HTTP applications incorporate a cache memory. This configures what is known as the **stateless client-cache-server protocol**: it is possible to define some of the responses to specific HTTP requests as cache-able, so the client can run **the same response for identical requests** in the future. However, the fact that the option exists doesn't mean it is the most recommended.
* There are four very important data transactions in any REST system and HTTP specification: **POST** (create), **GET** (read and consult), **PUT** (edit) and **DELETE**.
* Bottom of Form
* **Objects in REST are always manipulated from the URI**. It is the URI and no other element that is the sole identifier of each resource in this REST system. The URI allows us to access the information in order to change or delete it, or for example to share its exact location with third parties.
* **Uniform interface**: to transfer data, the REST system applies specific actions (POST, GET, PUT and DELETE) on the resources, provided they are identified with a URI. This makes it easier to obtain a uniform interface that systematizes the process with the information.
* **Layer system**: hierarchical architecture between the components. Each layer has a functionality within the REST system.
* **Use of hypermedia**: hypermedia is a term coined by [**Ted Nelson**](http://hyperland.com/) in 1965 and is an extension of the concept of hypertext. This concept, taken to web page development, is what allows the user to browse the set of objects through HTML links. In the case of a REST API, the concept of hypermedia explains the capacity of an app development interface to provide the client and the user with the adequate links to run specific actions on the data.
* All REST APIs must have the [**HATEOAS**](https://en.wikipedia.org/wiki/HATEOAS) (Hypermedia As The Engine Of Application State) principle to be genuine. This principle is what ensures that each time a request is made to the server and it returns a response, part of the information it contains will be the browsing hyperlinks associated to other client resources.

1. **Explain the advantages of RESTful web services?**

* **Separation between the client and the server**:

The REST protocol totally separates the user interface from the server and the data storage. This has some advantages when making developments. For example, it improves the portability of the interface to other types of platforms, it increases the scalability of the projects, and allows the different components of the developments to be evolved independently.

* **Visibility, reliability and scalability**:

The separation between client and server has one evident advantage, and that is that each development team can scale the product without too much problem. They can migrate to other servers or make all kinds of changes in the database, provided the data from each request is sent correctly. The separation makes it easier to have the front and the back on different servers, and this makes the apps more flexible to work with.

* **The REST API is always independent of the type of platform or languages**:

The REST API always adapts to the type of syntax or platforms being used, which gives considerable freedom when changing or testing new environments within the development. With a REST API you can have PHP, Java, Python or Node.js servers. The only thing is that it is indispensable that the responses to the requests should always take place in the language used for the information exchange, normally XML or JSON.