503073

WEB PROGRAMMING & APPLICATIONS

LECTURE 09 – Web Services

Instructor: Mai Van Manh

January 16, 2024

OUTLINE

1. Introduction
2. Restful Web Service
3. Create Restful Web Service in PHP
4. Consume Restful Web Service

Introduction

Introduction to Web Service

- A Web Service can be defined by following ways:
  - The method of communication between two devices over the network.
  - A software that makes itself available over the internet and uses a standardized data messaging system (XML/JSON).
  - It is a software system for the interoperable machine to machine communication.
  - It is a collection of standards or protocols for exchanging information between two devices or application.

Introduction

- Cross platform is one of the main characteristic of a web service.

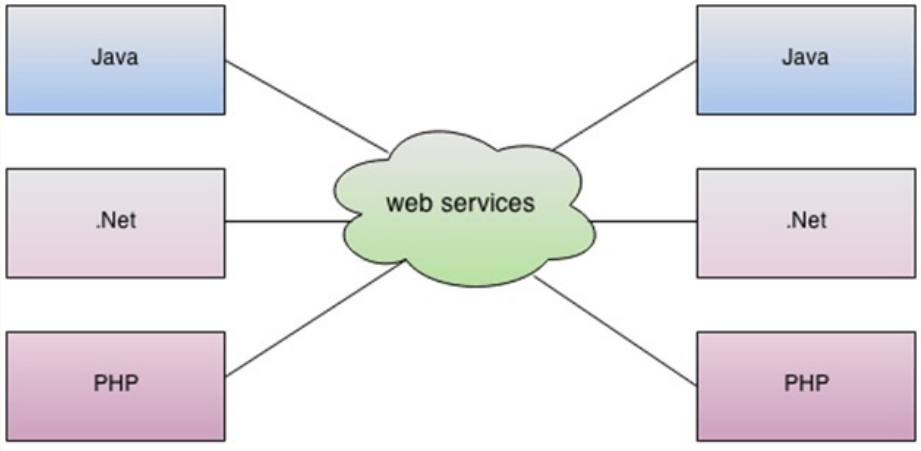


Image: www.javatpoint.com

Web Service

- To summarize, a complete web service is, any service that:
  - Is available over the Internet or private (intranet) networks
  - Uses a standardized XML or JSON messaging system
  - Is not tied to any one operating system or programming language
  - Is self-describing via a common XML grammar or JSON
  - Is discoverable via a simple find mechanism

Web Service Types

- Web services can be categorized as two types:
  1. SOAP web services (XML)
  2. RESTful web services (JSON)

SOAP Web Serivces

SOAP Web Service Components

1. SOAP - Simple Object Access Protocol
2. WSDL - Web Services Description Language
3. UDDI - Universal Description, Discovery and Integration

SOAP

- Here is what a SOAP message consists of:
  - a root element known as the <Envelope> element.
  - The <envelope> is divided into 2 parts: header and body
  - The header contains the routing data which is basically the information which tells the XML document to which client it needs to be sent to.
  - The body contains the actual message.

SOAP Message

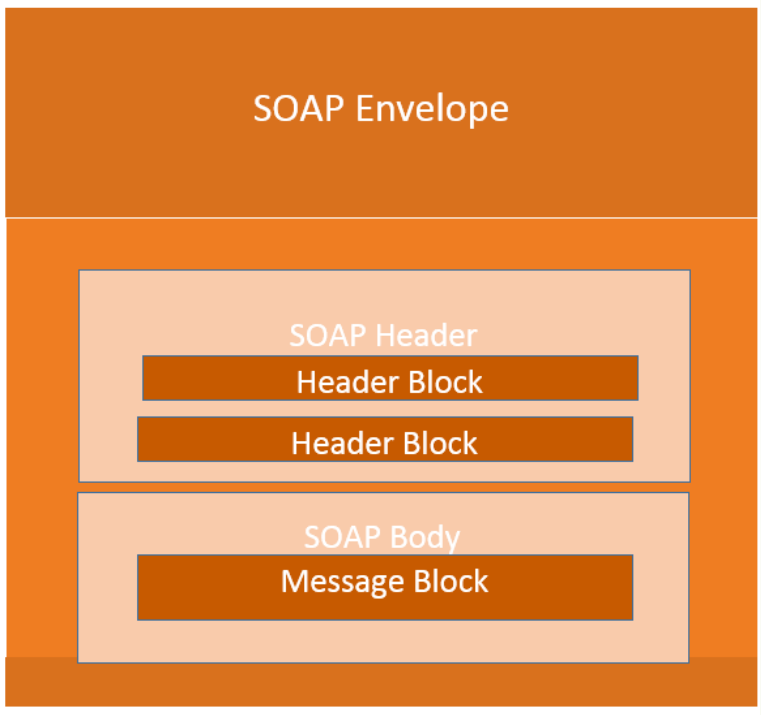


Image: www.guru99.com

SOAP Request Message

```
POST /HomeService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/Add"


<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/       2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <Add
            xmlns="http://tempuri.org/">
            <a>int</a>
            <b>int</b>
        </Add>
    </soap:Body>
</soap:Envelope>
```

HTTP Request

SOAP Response Message

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/       2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <AddResponse
            xmlns="http://tempuri.org/">
            <AddResult>int</AddResult>
        </AddResponse>
    </soap:Body>
</soap:Envelope>
```

HTTP Response

WSDL - Web Services Description Language

- The WSDL file is an XML-based file which tells the client application what the web service does.
- By using the WSDL document, the client application would be able to understand where the web service is located and how it can be utilized.
- In short, a WSDL file contains:
  - The location of the web service.
  - The methods which are exposed by the web service.

Universal Description, Discovery & Integration

- UDDI is a directory service.
- Web services can register with a UDDI and make themselves available through it for discovery.

SOAP Web Service Pros and Cons

- Pros:
  - WS Security: SOAP defines its own security known as WS Security.
  - Language and Platform independent: SOAP web services can be written in any programming language and executed in any platform.
- Cons:
  - Slow: SOAP uses XML format that must be parsed to be read.
  - WSDL dependent: SOAP uses WSDL and doesn't have any other mechanism to discover the service

RESTful Web Serivces

Restful Web services

- REST stands for REpresentational State Transfer.
- Web services based on REST Architecture are known as RESTful web services.
- In REST Architecture everything is a resource.
- These webservices uses HTTP methods to implement the concept of REST architecture.
- A RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation such as JSON and set of HTTP Methods.

Restful Architecture

- A REST server simply provides access to resources.
- REST client accesses and modifies the resources.
- 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.

Restful Architecture

- The key elements of a RESTful implementation are as follows:
  1. Resources
  2. Messages
  3. Request Verbs
  4. Request Headers
  5. Request Body
  6. Response Body
  7. Response Status codes

Resources

- REST architecture treats every content as a resource.
- Resources can be Text, Html, Images, Videos...
- REST Server simply provides access to resources and REST client accesses and modifies the resources.
- The most popular representations of resources are XML and JSON.

Resources Examples

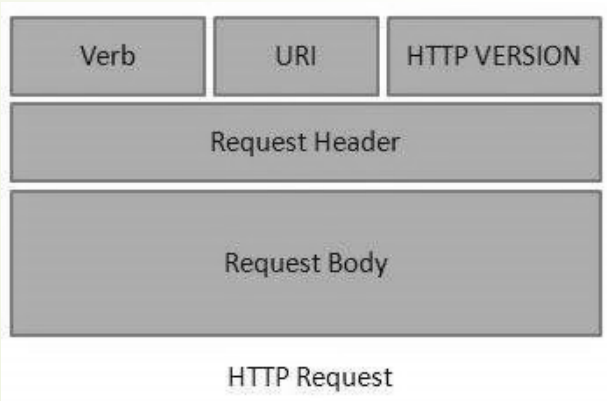
- XML and JSON representation of Rest resource.
- A resource in REST is a similar Object in OOP or is like an Entity in a Database.

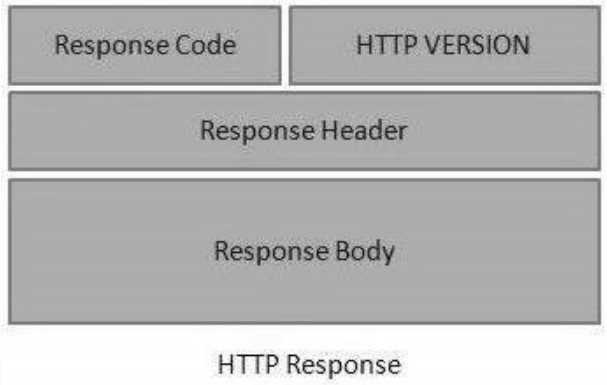
```xml
<user>
    <id>1</id>
    <name>Mahesh</name>
    <profession>Teacher</profession>
</user>
```

```json
{
    "id":1,
    "name":"Mahesh",
    "profession":"Teacher"
}
```

REST Message

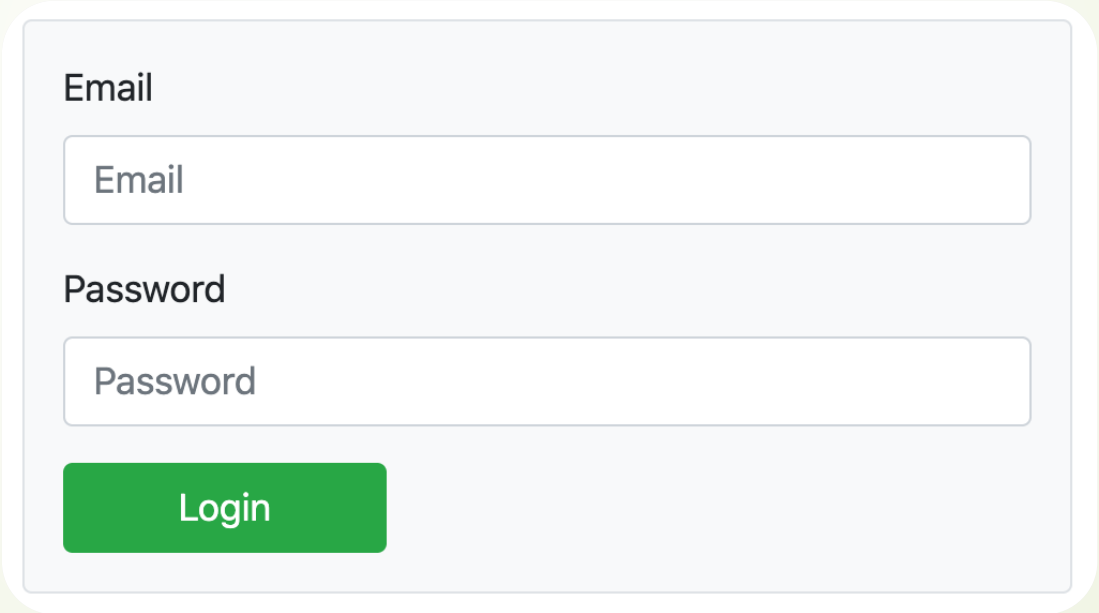
- RESTful Web Services make use of HTTP protocols as a medium of communication between client and server.
- A client sends a message in form of a HTTP Request and the server responds in the form of an HTTP Response.





REST Message Example

- Let's take the following HTML login form for demonstration purpose.



HTTP Request

- Verb – Indicates the HTTP methods such as GET, POST, DELETE, PUT, etc.
- URI – Uniform Resource Identifier (URI) to identify the resource on the server.
- HTTP Version – Indicates the HTTP version. For example, HTTP v1.1.
- Request Header – Contains metadata for the HTTP Request message as key-value pairs. For example, client (or browser) type, format supported by the client, format of the message body, cache settings, etc.
- Request Body – Message content or Resource representation.

HTTP Request

- The following HTTP Request message will be sent when we submit the previous HTML form.

telnet localhost 80

```
1 POST /login.php HTTP/1.1
2 Host: localhost
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 36
5
6 email=mvmanh%40gmail.com&pass=123456
```

HTTP Request

- Verb – POST URI – /login.php HTTP Version – HTTP/1.1.
- Request Header:
  - Content-Type: application/x-www-form-urlencoded
  - Content-Length: 36
- Request Body:
  - email=mvmanh%40gmail.com&pass=123456

telnet localhost 80

```
1 POST /login.php HTTP/1.1
2 Host: localhost
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 36
5
6 email=mvmanh%40gmail.com&pass=123456
```

HTTP Response

- Status/Response Code – Indicates the Server status for the requested resource. For example, 404 means resource not found and 200 means response is ok.
- HTTP Version – Indicates the HTTP version. For example HTTP v1.1.
- Response Header – Contains metadata for the HTTP Response message as keyvalue pairs. For example, content length, content type, response date, server type, etc.
- Response Body – Response message content or Resource representation.

HTTP Response

- The following HTTP Response message will be sent back.

telnet localhost 80

```
1 HTTP/1.1 200 OK
2 Date: Sun, 18 Oct 2020 11:55:11 GMT
3 Server: Apache/2.4.41 (Unix) PHP/7.3.11
4 X-Powered-By: PHP/7.3.11
5 Content-Length: 86
6 Content-Type: application/json
7
8 {"code":0,"data":{"full_name":"Mai Van Manh","avatar":"http:\/\/mvm.com\/avatar.jpg"}}
```

Resources Addressing

- Each resource in REST architecture is identified by its URI.

  <protocol>://<service-name>/<ResourceType>/<ResourceID>

- For examples:
  - http://shop.com/api/products/
  - http://shop.com/api/products /A381
  - http://shop.com/api/products?page=3

Restful Verbs

- Given REST Url: http://company.com/products
  - POST – Create a new product
  - GET - Get a list of all products
  - PUT - Update all product at a time
  - DELETE - Delete all product at a time

Restful Verbs

- Given REST Url: http://company.com/products/A381
  - POST: (not applicable)
  - GET - Get detail information of the product whose id = A381
  - PUT - Update product information whose id = A381
  - DELETE – Delete product whose id = A381

RESTFul Webservice

No.	URI	HTTP Method	POST body	Result
1	/UserService/users	GET	empty	Get list of all the users.
2	/UserService/addUser	POST	JSON String	Add details of new user.
3	/UserService/getUser/:id	GET	empty	Show details of a user.

Restful Advantages

- Language and Platform independent: RESTful web services can be written in any programming language and executed in any platform.
- Fast: RESTful Web Services are fast because there is no strict specification like SOAP. It consumes less bandwidth and resource.
- Can use SOAP: RESTful web services can use SOAP web services as the implementation.
- Permits different data format: RESTful web service permits different data format such as Plain Text, HTML, XML and JSON.

RESTful Web Serivce in PHP

Project Overview

- We will create a simple REST service that allows user to access Product resources:
  - List of all products
  - Add new products
  - Update a product
  - Delete a product

Database Structure

- We will create a database with only one table: Product.

#	Column	Data type	Description
1	id	integer	Primary key, auto
2	name	varchar(128)	
3	price	int	
4	desc	Varchar(255)	

REST Endpoint setup

- For the sake of simplicity, we'll use different php file for each operations.

Endpoint	Description
all_products.php	Returns all products
add_product.php	Add a new product
update_product.php	Update an existing product
delete_product.php	Delete a product

List All Products

- URL: http://localhost/all_products.php
- Method: GET
- Body: None
- Sample output:


```
{
    "code": 0,
    "message": "Đọc sản phẩm thành công",
    "data": [
        {
            "id": 1,
            "name": "iPhone X",
            "price": 1000,
            "desc": ""
        },
        {
            "id": 2,
            "name": "iPhone 11 Pro MAX",
            "price": 1200,
            "desc": ""
        }
    ]
}
```

Add New Product

- URL: http://localhost/add_product.php
- Method: POST
- Body: {"name":"iPad 8 ","price":599,"description":"Used"}
- Sample output:
  - {"code":1,"message":"Giá sản phẩm không hợp lệ"}
  - {"code":0,"message":"Thêm thành công", "data": 4}

Update Product

- URL: http://localhost/update_product.php
- Method: POST or PUT
- Body: {“id”: 5, "name":"iPad 8 ","price":599,"description":”Used"}
- Sample output:
  - {"code":1,"message":”Mã sản phẩm không tồn tại"}
  - {"code":0,"message":”Cập nhật thành công”}

Delete Product

- URL: http://localhost/delete_product.php
- Method: POST or DELETE
- Body: {“id”: 5}
- Sample output:
  - {"code":1,"message":"Mã sản phẩm không tồn tại"}
  - {"code":0,"message":"Xóa sản phẩm thành công"}

Database Setup

db.php

```php
<?php
    $host = '127.0.0.1';
    $user = 'mvmanh';
    $pass = '123456';
    $db = 'week9_web_service';

    $conn = new mysqli($host, $user, $pass, $db);
    if ($conn->connect_error) {
        die('Không thể kết nối database');
    }
?>

```

all_products.php

db.php

```php
require_once('db.php');
header('Content-Type: application/json');

$sql = 'select * from product';
$result = $conn->query($sql);

if (!$result) {
    echo json_encode(array('code' => 1, 'message' => $conn->error));
}

$data = $result->fetch_all();

echo json_encode(array('code' => 1, 'message' => 'Đọc thành công',
    'data' => $data));
```

all_products.php


```
{
    "code": 1,
    "message": "Đọc thành công",
    "data": [
        [
            "1",
            "iPhone X",
            "1000",
            "iPhone 10 xách tay"
        ],
        [ … ], // 4 items
        [ … ], // 4 items
        [ … ] // 4 items
    ]
}
```

add_products.php

- First, read input as json object and validate input parameters.

add_products.php

```php
$input = json_decode(file_get_contents('php://input'));

if (!property_exists($input,'name') ||
    !property_exists($input,'price') ||
    !property_exists($input,'description')) {
    die(json_encode(array('code' => 1, 'message' =>
    'Thiếu thông tin đầu vào')));
}

if (empty($input->name) ||
    empty($input->price) ||
    empty($input->description)) {
    die(json_encode(array('code' => 2, 'message' =>
    'Thông tin đầu vào không hợp lệ')));
}
```

add_products.php

- You can also accept input paramters as form fields (x-www-forrm-urlencoded)

add_products.php

```php
if (!isset($_POST['name']) || !isset($_POST['price']) ||
    !isset($_POST['description'])) {

    die(json_encode(array('code' => 1, 'message' =>
    'Thiếu thông tin đầu vào')));
}

$name = $_POST['name'];
$price = $_POST['price'];
$desc = $_POST['description'];

if (empty($name) || empty($price) || empty($desc)) {

    die(json_encode(array('code' => 2, 'message' =>
    'Thông tin đầu vào không hợp lệ')));
}
```

add_products.php (cont)

- After checking the input, we use Prepared Statement to insert new record.

add_products.php

```php
require_once('db.php');

$sql = 'insert into product(name, price, description) values(?,?,?)';
$stm = $conn->prepare($sql);

$stm->bind_param('sis', $name, $price, $desc);

if (!$stm->execute()) {
    die(json_encode(array('code' => 3, 'message' => $stm->error)));
}

$id = $stm->insert_id;
die(json_encode(array('code' => 0, 'message' => 'Chèn thành công',
    'data' => $id)));
```

add_products.php (cont)

- **Sample output**: “5” is the id of the inserted product

```
{
    "code": 0,
    "message": "Chèn thành công",
    "data": 5
}
```

update_product.php

update_products.php

```php
$sql = 'update product set name = ?, price = ?,
            description = ? where id = ?';

$stm = $conn->prepare($sql);

$stm->bind_param('sisi', $name, $price, $desc, $id);

if (!$stm->execute()) {
    die(json_encode(array('code' => 3, 'message' => $stm->error)));
}

die(json_encode(array('code' => 0, 'message' => 'Cập nhật thành công')));
```

delete_product.php

delete_product.php

```php
$sql = 'delete from product where id = ?';
$stm = $conn->prepare($sql);

$stm->bind_param('i', $id);

if (!$stm->execute()) {

    die(json_encode(array('code' => 3, 'message' => $stm->error)));
}

die(json_encode(array('code' => 0, 'message' =>
        'Xóa sản phẩm thành công')));
```

Test Restful Web Service

REST Client

- Postman
- Advanced REST Client

Ajax Client

- XMLHttpRequest
- jQuery ajax
- fetch()

Read Products with XmlHttpRequest

XML HTTP Request

- All modern browsers have a built-in XMLHttpRequest object to request data from a server.
- The XMLHttpRequest object can be used to:
  1. Update a web page without reloading the page
  2. Request data from a server - after the page has loaded
  3. Receive data from a server - after the page has loaded
  4. Send data to a server - in the background

XML HTTP Request

- All modern browsers have a built-in XMLHttpRequest object to request data from a server.
- The XMLHttpRequest object can be used to:
  1. Update a web page without reloading the page
  2. Request data from a server - after the page has loaded
  3. Receive data from a server - after the page has loaded
  4. Send data to a server - in the background

XML HTTP Request

- Create an XMLHttpRequest object, set the response type to 'json'.

```
1 let ajax = new XMLHttpRequest();
2 ajax.responseType = 'json';
```

XML HTTP Request

- Add an event handler for 'readystatechange' event. This is where we should handle and display the result.

```
1 ajax.addEventListener('readystatechange', () => {
2     if (ajax.readyState === 4 && ajax.status === 200) {
3         let json = ajax.response;
4         console.log(json.code);
5         json.data.forEach(i => console.log(i));
6     }
7 });
```

XML HTTP Request

- Use open() and send() method to send the request.

```
ajax.open('GET', 'http://localhost/all_products.php',
          'true'); // true có nghĩa là asynchronous
ajax.send();
// sau khi gọi send thì readystatechange sẽ hoạt động
```

XML HTTP Request

- Full source code

```
let ajax = new XMLHttpRequest();
ajax.responseType = 'json';

ajax.addEventListener('readystatechange', () => {
    if (ajax.readyState === 4 && ajax.status === 200) {
        let json = ajax.response;
        console.log(json.code);
        json.data.forEach(i => console.log(i));
    }
});
ajax.open('GET', 'http://localhost/api/all_products.php', 'true');
ajax.send();
```

XML HTTP Request

- Sample output

```
▼ (4) ["2", "iPhone 11 Pro", "1100", "Hàng second hand còn bảo hành"]
    0: "2"
    1: "iPhone 11 Pro"
    2: "1100"
    3: "Hàng second hand còn bảo hành"
    length: 4
  ▶ __proto__: Array(0)
▶ (4) ["3", "iPhone 11 Pro MAX", "1200", "Chưa đập hộp"]
▶ (4) ["4", "iPhone 12", "1500", "Mẫu mới nhất vừa ra mắt"]
▶ (4) ["5", "Apple Watch 5S", "399", "Hàng quá cũ"]
```

Add Product with XmlHttpRequest (using form)

Create new Product with XML HTTP Request

- We can send POST request with raw json or form encoded.

```
let form = new FormData();
form.set('name','iPad Pro 2020');
form.set('price',1500);
form.set('description','New Model');

let params = new URLSearchParams(form).toString();
// name=iPad+Pro+2020&price=1500&description=New+Model

```

Create new Product with XML HTTP Request

```
let ajax = new XMLHttpRequest();

ajax.open('POST', 'http://localhost/add_product.php', 'true');
ajax.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
ajax.responseType = 'json';

ajax.addEventListener('readystatechange', () => {
    if (ajax.readyState === 4 && ajax.status === 200) {
        console.log(ajax.response);
    }
});

ajax.send(params);
```

Create new Product with XML HTTP Request

▶ {code: 0, message: "Chèn thành công", data: 11}

Add Product with XmlHttpRequest (using JSON)

Send JSON with XML HTTP Request

- First, create JS object which will be inserted

```
let product = {
    name: 'iMac 2020',
    price: 2500,
    description: 'New model'
};
```

Send JSON with XML HTTP Request

- Set 'application/json' as content type.

```
let xhr = new XMLHttpRequest();
xhr.responseType = 'json';
xhr.open('POST', 'http://localhost/api/add_product.php', true);
xhr.setRequestHeader('Content-Type','application/json');
```

Send JSON with XML HTTP Request

- Convert JS object to JSON string then send it

```
1 xhr.addEventListener('readystatechange', () => {
2     if (xhr.readyState === 4 && xhr.status === 200) {
3         console.log(xhr.response);
4     }
5 });
6
7 xhr.send(JSON.stringify(product));
```

Demo