Web Services Fundamentals and Implementation in PHP

Hands-on Workshop Organized by Microsoft Learn Student Club (MLSC), BVM
Department of Information Technology

Prof. Nayankumar M. Mali
Department of Information Technology, ADIT

17 September 2025
3:00 – 5:00 PM
Room B-201
Github Link .

Welcome Agenda

Welcome to the Hands-on Workshop on **Web Services in PHP**

Organized by MLSC BVM & Department of IT

Agenda

1. Fundamentals of Web Services
2. REST Architecture & Implementation
3. SOAP Protocol & Implementation
4. GraphQL Basics
5. Industrial Use Cases
6. Hands-on Coding Demo
7. Q & A

Web Services Fundamentals

What is a Web Service?

- Standardized way for applications to communicate over the web
- Platform & language independent
- Uses open protocols: HTTP/HTTPS, XML, JSON
- Enables interoperability across diverse systems

Key Characteristics

- Platform and language neutral
- Standard protocols (HTTP, XML, SOAP, REST, JSON)
- Self-describing (WSDL or OpenAPI)
- Discoverable (UDDI – rarely used today)

Types of Web Services

Type	Description
SOAP	Protocol-based, XML-heavy, enterprise-grade
REST	Lightweight architectural style using HTTP methods
GraphQL	Query language allowing exact data retrieval

Example

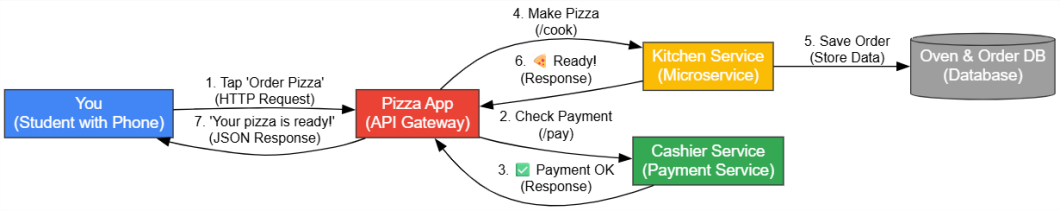

Figure 1: Caption

REST Fundamentals

REST Overview

- Representational State Transfer (REST)
- Uses standard HTTP verbs: GET, POST, PUT, DELETE
- Stateless communication
- JSON is most common data format

Sample REST Endpoint

```
GET /api/users
Response:
{
  "users": ["Alice","Bob"]
}
```

HTTP Methods Mapping

GET	Retrieve resource
POST	Create resource
PUT	Update resource
DELETE	Remove resource

REST in Native PHP – Part 1

```php
<?php
header('Content-Type: application/json');
$method = $_SERVER['REQUEST_METHOD'];
$path   = $_SERVER['PATH_INFO'] ?? '/';

switch($method) {
  case 'GET':
    if($path === '/users') {
        echo json_encode(['users'=>['Alice','Bob']]);
    }
    break;
```

REST in Native PHP – Part 2

```
  case 'POST':
    if($path === '/users') {
       $input = json_decode(file_get_contents('php://input'),
          true);
       // Save to DB
       echo json_encode(['status'=>'User created','data'=>
          $input]);
    }
    break;
  default:
    http_response_code(405);
    echo json_encode(['error'=>'Method not allowed']);
}
```

Laravel REST Example

```php
// routes/api.php
Route::apiResource('users', UserController::class);

// app/Http/Controllers/UserController.php
public function index() {
    return response()->json(User::all());
}

public function store(Request $req) {
    $user = User::create($req->all());
    return response()->json($user,201);
}
```

SOAP Fundamentals

SOAP Overview

- Simple Object Access Protocol
- Strict XML messaging standard
- WSDL for describing services
- Widely used in enterprise systems

SOAP Server Example

```
<?php
class UserService {
  public function getUser($id) {
    return "User ID: $id";
  }
}

$options = ['uri' => 'http://localhost/soap'];
$server = new SoapServer(null, $options);
$server->setClass('UserService');
$server->handle();
```

SOAP Client Example

```php
<?php
$client = new SoapClient(null, [
 'location' => "http://localhost/server.php",
 'uri'      => "http://localhost/soap"
]);
echo $client->getUser(123);
```

GraphQL Overview

GraphQL Basics

- Query language for APIs
- Client specifies exact data requirements
- Single endpoint

GraphQL PHP Example

```
$queryType = new ObjectType([
 'name' => 'Query',
 'fields' => [
   'user' => [
     'type' => $userType,
     'args' => ['id' => Type::string()],
     'resolve' => fn($root, $args)
        => ['id' => $args['id'], 'name' => 'John Doe']
   ]
 ]
]);
```

Industrial Use Cases

Industry Applications

1. E-Commerce: product catalog sync
2. Banking/Finance: transaction APIs
3. Healthcare: patient record exchange (FHIR/HL7)
4. Logistics: real-time shipment tracking
5. SaaS: CRM/ERP integrations

E-Commerce Example

- Tech: REST API in Laravel/Symfony
- Example: Magento 2 inventory order management

Banking & Finance

- SOAP for legacy core banking systems
- REST for mobile banking apps

Healthcare Systems

- REST/JSON with OAuth2 security
- Enables hospital-to-clinic interoperability

Logistics & Tracking

- REST + Webhooks
- FedEx/UPS tracking APIs

Mobile Backend Services

- Serve data to iOS/Android
- Laravel + JWT Auth

Microservices Architecture

- Decoupled services for auth, payment, notifications
- Slim/Laravel + RabbitMQ/gRPC

Security Considerations

API Security Essentials

- Enforce HTTPS
- Input validation sanitization
- Authentication: JWT, OAuth2, API Keys
- Rate limiting & throttling
- CORS configuration

Best Practices

API Best Practices

- Meaningful HTTP status codes
- Versioned endpoints: /api/v1/...
- Consistent naming (camelCase or snake_case)
- Proper logging and monitoring
- Cache where possible (Redis/Memcached)

Hands-on Demo Plan

Workshop Activity

1. Build a RESTful API in Native PHP
2. Test using Postman
3. Extend with authentication

Tools to Install

- PHP 8+, Composer
- Web server: Apache/Nginx
- Postman or Insomnia
- Optional: Laravel for advanced demo

Resources

Recommended References

- PHP Manual: SOAP Extension
- Laravel API Documentation
- GraphQL PHP
- OpenAPI Specification

Closing

Summary

- PHP supports REST, SOAP, GraphQL
- Wide industrial adoption: e-commerce, finance, healthcare
- Security and best practices are critical

Thank You!

Questions?