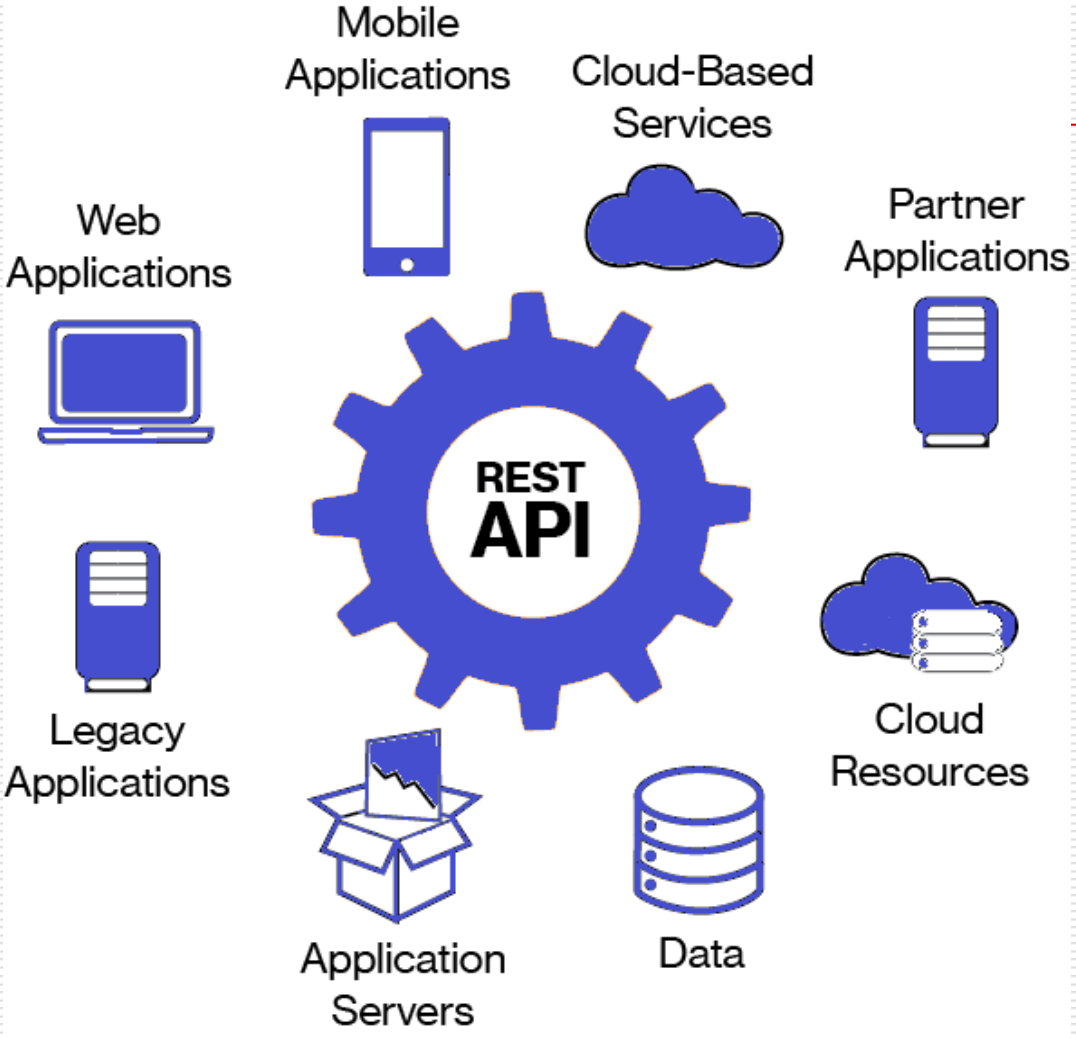
Web-технологии и web-программирование

Web-сервисы и API



Протокол HTTP

- HyperText Transfer Protocol — протокол прикладного уровня.
- Ориентирован на модель обмена "клиент-сервер".
- Клиент и сервер обмениваются фрагментами данных, которые называются *HTTP-сообщениями*.
- Сообщения, отправляемые клиентом серверу, называют запросами, а сообщения, отправляемые сервером клиенту — откликами.
- Сообщение может состоять из двух частей: заголовка и тела.
  - **Заголовок** содержит служебную информацию, необходимую для обработки тела сообщения или управления обменом. Заголовок состоит из директив заголовка, которые обычно записываются каждая на новой строке.
  - **Тело сообщения** не является обязательным, в отличие от заголовка сообщения. Оно может содержать текст, графику, аудио- или видеоинформацию.
  - Тело от заголовка отделяется *пустой строкой*.

Протокол HTTP

HTTP-запрос:

- **GET / HTTP/1.0**
  Accept: text/html
  пустая строка

Отклик:

- **HTTP/1.0 200 OK**
  Date: Fri, 24 Jul 1998 21:30:51 GMT
  Server: Apache/1.2.5
  Content-type: text/html
  Content-length: 21345
  пустая строка
  <HTML>
  ...
  </HTML>
- Часть информации заголовка HTTP-запроса преобразуется сервером в переменные окружения, которые доступны для анализа CGI-скриптом.
- Тело запроса становится доступным скрипту через поток стандартного ввода.

Протокол HTTP: метод доступа

- является самой главной директивой HTTP-запроса
- указывается первым словом в первой строке запроса.

Метод GET

- применяется клиентом при запросе к серверу по умолчанию.
- в заголовке HTTP-запроса клиент сообщает адрес ресурса (URL), который он хочет получить, версию протокола HTTP, поддерживаемые им MIME-типы документов, версию и название клиентского программного обеспечения.
- тело в запросе не передается.
- В ответ сервер сообщает версию HTTP-протокола, код возврата, тип содержания тела сообщения, размер тела сообщения и ряд других необязательных директив HTTP-заголовка.
- Сам ресурс, обычно HTML-страница, передается в теле отклика.

Протокол HTTP: метод доступа

Метод POST

- альтернатива методу GET.
- при обмене данными по методу POST в запросе клиента присутствует тело HTTP-сообщения.
- Это тело может формироваться из данных, которые вводятся в HTML-форме, или из присоединенного внешнего файла.
- В отклике, как правило, присутствует и заголовок, и тело HTTP-сообщения.

SOA - сервисно-ориентированная архитектура

- компонентная модель, состоящая из отдельных функциональных модулей приложений, называемых **сервисами**, имеющих определенные согласно некоторым общим правилам интерфейсы и механизм взаимодействия между собой.

SOA **не является** технологией или набором технологий, это - концепция, абстрактное представление реализации информационных систем с помощью сервисов безотносительно конкретных технологий.

Web- сервис

Современное коммерческое предприятие использует большие многофункциональные информационные системы (ERP, CRM, SCM и т. п.), часто несколько одновременно.

Сервис (service) - ресурс, реализующий бизнес-функцию, обладающий следующими свойствами:

является повторно используемым;
определяется одним или несколькими явными технологически-независимыми интерфейсами;
слабо связан с другими подобными ресурсами и может быть вызван посредством коммуникационных протоколов, обеспечивающих возможность взаимодействия ресурсов между собой.

Сторонние общедоступные API чаще всего отдают данные в XML или JSON.

Взаимодействие между компонентами SOA

Различают следующие **три основных архитектурных компонента** сервисно-ориентированной архитектуры:

- **пользователь** сервиса: приложение, программный модуль либо сервис, осуществляющий поиск и вызов необходимого сервиса из реестра сервисов по описанию сервиса, а также использующий сервис, предоставляемый провайдером сервиса, в соответствии с интерфейсом сервиса;

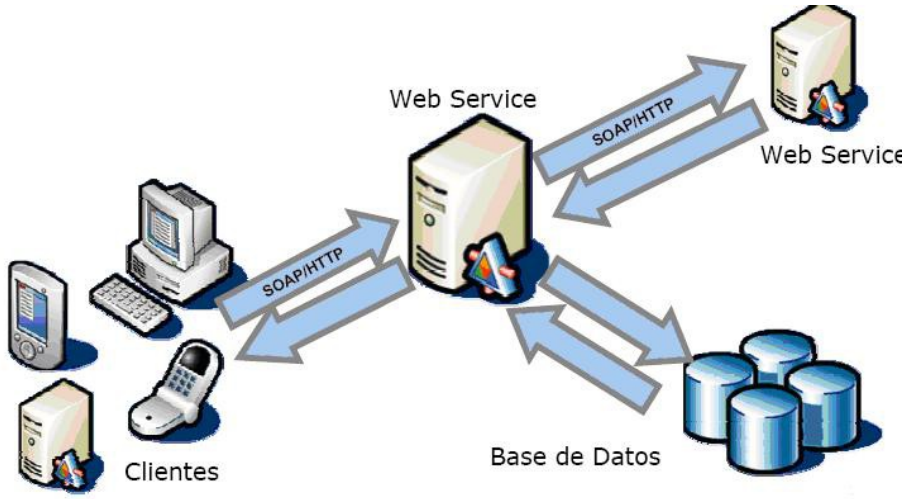
- **провайдер** сервиса: приложение, программный модуль либо сервис, осуществляющий реализацию сервиса в виде веб-сервиса, прием и исполнение запросов пользователей сервиса, а также публикацию сервиса в реестре сервисов;

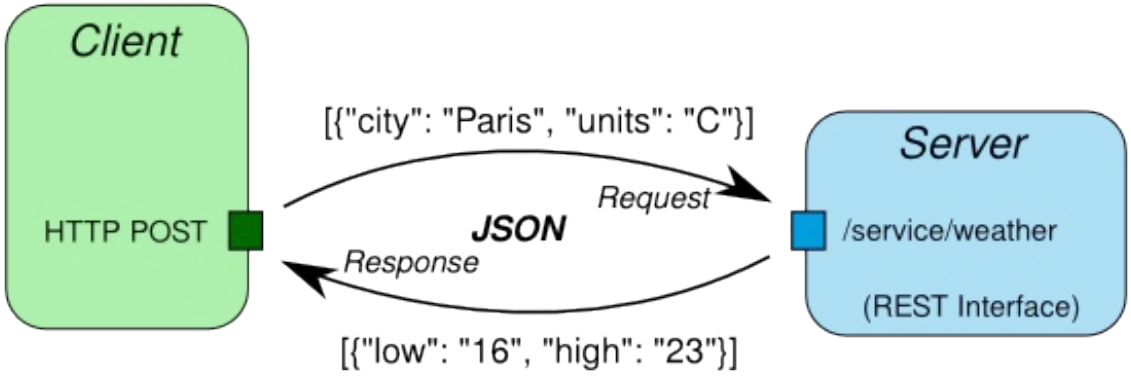
- **реестр** сервисов: библиотека сервисов, предоставляющая пользователям сервиса средства поиска и вызова необходимого сервиса и принимающая запросы провайдеров сервисов на публикацию сервисов.

Взаимодействие между компонентами SOA (на примере SOAP-XML)

Технологии web-сервисов

1. XML Web Service на протоколе SOAP.
2. RESTful API приложение.





RESTful API

Application **P**rogramming **I**nterface -
интерфейс, который позволяет разработчикам использовать готовые блоки для построения своего приложения.

RESTful API - может отдавать данные в формате, отличном от стандартного HTML, благодаря чему им удобно пользоваться при написании собственных приложений.

REST (Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети.

Принципы RESTful

1. Независимость от состояния: данные, возвращаемые определенным вызовом API, не должны зависеть от вызовов, сделанных ранее.
2. Кэширование: предоставление клиенту информации о том, что ответ сервера может быть кэширован на определенный период времени и использоваться повторно без новых запросов к серверу.
3. Клиент – серверное разделение и единый интерфейс: клиенту не следует знать о том, какая СУБД используется на сервере или сколько серверов в данный момент обрабатывают запросы и прочие подобные вещи.

REST запросы

POST - для создания ресурса на сервере.

```
POST /users HTTP/1.1
Host: myserver
Content-Type: application/xml
<?xml version="1.0"?>
<user><name>Robert</name></user>
```

GET - для извлечения ресурса.

```
GET /users/Robert HTTP/1.1
Host: myserver
Accept: application/xml
```

PUT - для изменения состояния ресурса или его обновления.

```
PUT /users/Robert HTTP/1.1
Host: myserver
Content-Type: application/xml
<?xml version="1.0"?>
<user>  <name>Bob</name></user>
```

DELETE - для удаления ресурса.

JSON - JavaScript Object Notation

- текстовый формат обмена данными, основанный на JavaScript. Пример (для api-схемы ВКонтакте):

Объект
{"id":210700286,
"first_name":"Lindsey",
"last_name":"Stirling"}

JSON-схема, описывающая этот объект:

```
{
"type": "object",
"properties": {
    "id": {"type":"integer", "description":"User ID"},
    "first_name": {"type":"string", "description":"First name"},
    "last_name": {"type":"string", "description":"Last name"}
    },
"required": ["id","first_name","last_name"],
"additionalProperties": false
}
```

Пример API приложения

1. База данных, таблица “Товар”:
'id','name','company','category','quantity','description','price'

2. PHP скрипт для подключения к базе данных:
database.php
class Database{...return $this->conn;}

3. PHP скрипт для получения данных о товаре:
product.php
class Product {... function read() { ...return $stmt; //результат запроса select по ID товара} }

Скрипт **read.php** для доступа к списку всех товаров:

```php
<?php
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
include_once 'database.php'; include_once 'product.php';
$database = new Database();          $db = $database->getConnection();
$product = new Product($db);         $stmt = $product->read();
$num = $stmt->rowCount();
if($num>0) {
    $products_arr=array();                  $products_arr["records"]=array();

    while ($row = $stmt->fetch(PDO::FETCH_ASSOC)){
        extract($row);
        $product_item=array("id" => $id, "name" => $name, "company" => $company,
          "category" => $category,  "quantity" => $quantity,
          "description" => html_entity_decode($description), "price" => $price);
        array_push($products_arr["records"], $product_item);
    }
    echo json_encode($products_arr, JSON_UNESCAPED_UNICODE);}
else{
    echo json_encode(
       array("message" => "No products found.")    );
}
?>
```

Обращение к API: вывод всех товаров

http://localhost/api/read.php


```
{
"records":[
        {
        "id":"1",
        "name":"Tahoe 3 Camo",
        "company":"Nordway",
        "category":"Трекинговая",
        "quantity":"3",
        "description":"Классическая 3-х местная палатка с вместительным передним тамбуром.",
        "price":"11000"
        },
        {
        "id":"2","name":"Artezian 3",
        "company":"Salewa",
        "category":"Кемпинговая",
        "quantity":"3",
        "description":"Палатка Salewa Artesian III - удобная палатка с высокой водонепроницаемостью,
                редназначенная для кемпинга. Вместимость палатки - 3 человека, геометрия купола -
                полусфера. Тент и дно кемпинговой палатки изготовлены из нейлона, все швы
                проклеены, вход во внутреннюю палатку оборудован москитной сеткой.",
        "price":"23000"
        },
        {
        "id":"3",
        "name":"Peak 2",
        "company":"Rock Land",
        "category":"Трекинговая",
        "quantity":"2",
        "description":"Для активного отдыха, туризма, рыбалки и охоты. Небольшой вес и компактная упаковка
                - хороший вариант для путешествий пешком. Два входа обеспечивают хорошую вентиляцию.",
        "price":"6000"
        }
        ]
}
```

read_one.php

```
<?php
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: GET");
header('Content-Type: application/json');
include_once 'database.php';
include_once 'product.php';

$database = new Database();
$db = $database->getConnection();

$product = new Product($db);
$product->id = isset($_GET['id']) ? $_GET['id'] : die();
$product->readOne();

$product_arr=array("id" => $product->id, "name" => $product->name,
      "company" => $product->company,
       "category" => $product->category,  "quantity" => $product->quantity,
       "description" => $product->description, "price" => $product->price);

print_r(json_encode($product_arr, JSON_UNESCAPED_UNICODE));
?>
```

header("Access-Control-Allow-Headers: access");
header("Access-Control-Allow-Credentials: true");

product.php: метод read_one()

...

```
function readOne(){
    $query = "SELECT
            c.name as category_name, p.id, p.name, p.description, p.price,
p.category_id, p.created
        FROM " . $this->table_name . " p WHERE  p.id = ? LIMIT 0,1";

    $stmt = $this->conn->prepare( $query );
    $stmt->bindParam(1, $this->id);
    $stmt->execute();
    $row = $stmt->fetch(PDO::FETCH_ASSOC);

    $this->name = $row['name'];
    $this->company = $row['company'];
    $this->category = $row['category'];
    $this->quantity = $row['quantity'];
    $this->description = $row['description'];
    $this->price = $row['price'];
}
```

Обращение к API: вывод товара по id

http://localhost/api/read.php?id=3


```
{
"id":"3",
"name":"Peak 2",
"company":"Rock Land",
"category":"Трекинговая",
"quantity":"2",
"description":"Для активного отдыха, туризма, рыбалки и охоты. Небольшой вес и компактная упаковка
        - хороший вариант для путешествий пешком. Два входа обеспечивают хорошую вентиляцию.",
"price":"6000"
}
```


Пример:
https://www.codeofaninja.com/2017/02/create-simple-rest-api-in-php.html