

# Talking to Other Websites and Services

---

## Chapter 11

# Data From Other Websites

---

- ⋮ **Databases** and **files** are 2 sources of data
- ⋮ another important external source of data: **other websites**.
- ⋮ Your program can be **client** or **server** (provider) of data to other websites
  
- ⋮ What is an **API?** (Application Programming Interface)
- ⋮ It is a **software Interface** that allows two applications to talk to each other.
- ⋮ **Example:**
  - When you use an **App** on your mobile phone, the App connects to the Internet and **sends data to a server**.
  - The **server** then **retrieves** that data, **interprets** it, performs the necessary **actions** and **sends it back** to your phone.
  - The App then **interprets that data** and **presents** you with the information you wanted in a readable way.

# Simple URL Access with File Functions

---

⋮ We can use `file_get_contents()` to access remote files by their URL

```
Did you know that <?= file_get_contents('http://numbersapi.com/09/27') ?>
```

⋮ Check yourself: [numbersapi.com/09/27](http://numbersapi.com/09/27)

# Simple URL Access with File Functions

---

⋮ `http_build_query()` is useful for building an API URL

- Give it an **associative array** of parameter **names** and **values** and it gives you back **a string of key=value pairs** joined by & and properly encoded—exactly what you need for a URL.

⋮ Some servers (e.g., usda.gov) need you to provide **a key to access an API**

```
define('NDB_API_KEY', '273bqhebrfkhuebf');  
  
$params = array('api_key' => NDB_API_KEY,  
                'q' => 'black pepper',  
                'format' => 'json');  
  
$url = "http://api.nal.usda.gov/ndb/search?" . http_build_query($params);
```

⋮ **The Built Query (URL):**

```
http://api.nal.usda.gov/ndb/search?  
api_key=j724nbefuy72n4&q=black+pepper&format=json
```

# JSON as response

---

After calling `file_get_contents($url)`, the API returns **JSON**

```
{
  "list": {
    "q": "black pepper",
    "sr": "27",
    "start": 0,
    "end": 1,
    "total": 1,
    "group": "",
    "sort": "r",
    "item": [
      {
        "offset": 0,
        "group": "Spices and Herbs",
        "name": "Spices, pepper, black",
        "ndbno": "02030"
      }
    ]
  }
}
```

# What is JSON?

---

⋮ JSON: JavaScript Object Notation

⋮ JSON is a **text format** for **storing** and **transporting** data

⋮ JSON is "**self-describing**" and easy to understand

⋮ Data is written in **JavaScript Object** notation

⋮ JSON Syntax is **derived from JavaScript** but JSON is **supported by many programming languages**.

⋮ [JSON Introduction \(w3schools.com\)](https://www.w3schools.com/js/json_intro.asp)

## JSON as response

---

- Pass the JSON response above to `json_decode()` to transform the [JSON into a PHP data structure](#) you can manipulate.

```
$params = array('api_key' => NDB_API_KEY,
                'q' => 'black pepper',
                'format' => 'json');

$url = "http://api.nal.usda.gov/ndb/search?" . http_build_query($params);
$response = file_get_contents($url);
$info = json_decode($response);

foreach ($info->list->item as $item) {
    print "The ndbno for {$item->name} is {$item->ndbno}.\n";
}
```

# JSON to a PHP Data Structure

---

⋮ The `json_encode()` function turns:

- JSON objects into PHP objects
- JSON arrays into PHP arrays
- The top-level item in the response is an object.

⋮ The NDB API call returns JSON because of the `format=json` query string parameter.

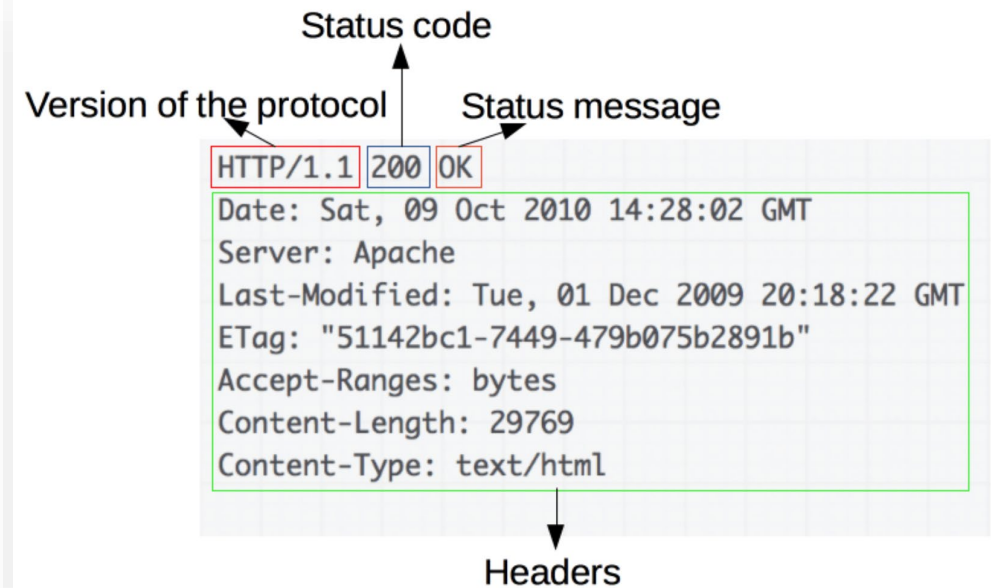
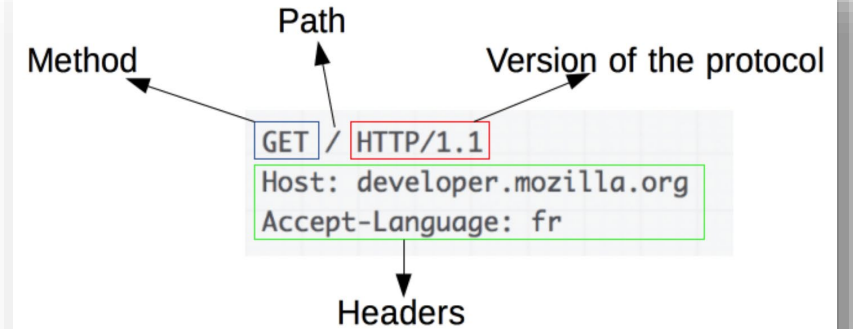
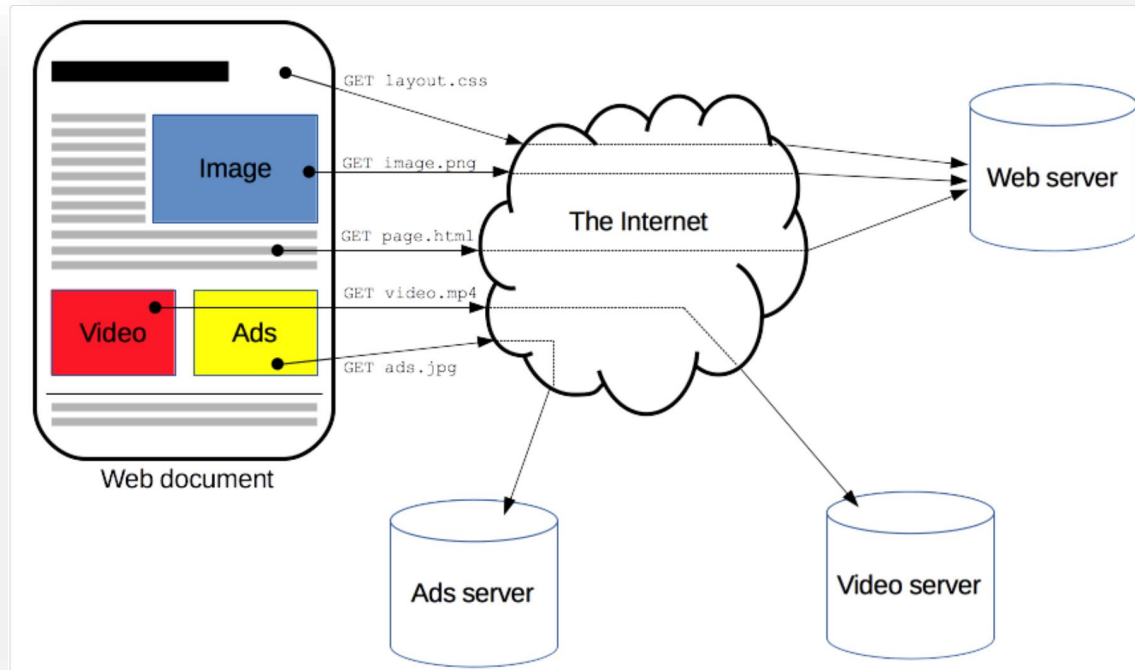
⋮ The API also supports specifying the response format by sending a Content-Type header.

- A header value of `application/json` tells the server to format the response as JSON.
- The context is also how you send a POST request



# HTTP Header

Every HTTP request or response has a header



# HTTP Status Codes

---

- ⋮ Informational responses (100 – 199)
- ⋮ Successful responses (200 – 299)
- ⋮ Redirection messages (300 – 399)
- ⋮ Client error responses (400 – 499)
- ⋮ Server error responses (500 – 599)
  
- ⋮ **More info:** <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

# Setting the Header for POST request

---

⋮ `stream_context_create()` function is used for **setting the header**

## Syntax

```
file_get_contents(path, include_path, context, start, max_length)
```

⋮ `include_path` is optional, but mandatory if you want to set context

- Set it to true or '1' if you want to search for the file in the include\_path (in php.ini) as well, else **set it to false**

⋮ To know more about **Context options**:

⋮ <https://www.php.net/manual/en/context.php>

```
// Just key and query term, no format specified in query string
$params = array('api_key' => NDB_API_KEY,
                'q' => 'black pepper');
$url = "http://api.nal.usda.gov/ndb/search?" . http_build_query($params);

// Options are to set a Content-Type request header
$options = array('header' => 'Content-Type: application/json');
// Create a context for an 'http' stream
$context = stream_context_create(array('http' => $options));

// Pass the context as the third argument to file_get_contents
print file_get_contents($url, false, $context);
```

# Sending a POST request

---

```
$url = 'http://php7.example.com/post-server.php';

// Two variables to send via POST
$form_data = array('name' => 'black pepper',
                  'smell' => 'good');

// Set the method, content type, and content
$options = array('method' => 'POST',
                'header' => 'Content-Type: application/x-www-form-urlencoded',
                'content' => http_build_query($form_data));

// Create a context for an 'http' stream
$context = stream_context_create(array('http' => $options));

// Pass the context as the third argument to file_get_contents.
print file_get_contents($url, false, $context);
```

# Using cURL

---

PHP's **cURL functions** using **libcurl** provides **powerful** HTTP requests and responses

Three steps:

- `curl_init()` : returns a handle
- `curl_setopt()` : set options
- `curl_exec()` : the request is retrieved

**CURLOPT\_RETURNTRANSFER** option asks cURL to return the response as a string, otherwise it is printed out.

```
<?php
```

```
$c = curl_init('http://numbersapi.com/09/27');  
// Tell cURL to return the response contents as a string  
// rather than printing them out immediately  
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);  
// Execute the request  
$fact = curl_exec($c);
```

```
?>
```

```
Did you know that <?= $fact ?>
```

# Using cURL

Two kinds of **errors** when using cURL:

- **Error from cURL itself:** not finding the host, or not being able to connect to the remote server
  - `curl_exec()` returns *false*
  - `curl_errno()` returns an error code
  - `curl_error()` returns the error message corresponding to the code
- **Error from the remote server:** the URL not found by the server, or it has a problem producing a response
  - **NOTE** cURL consider this case successful because the server returns something
  - You need to check the HTTP response code
  - `curl_getinfo()` returns an *array* about the request. One in that array is the *HTTP response code*.

```
// A pretend API endpoint that doesn't exist
$c = curl_init('http://api.example.com');
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);
$result = curl_exec($c);
// Get all the connection info, whether or not it succeeded
$info = curl_getinfo($c);

// Something went wrong with the connection
if ($result === false) {
    print "Error #" . curl_errno($c) . "\n";
    print "Uh-oh! cURL says: " . curl_error($c) . "\n";
}

// HTTP response codes in the 400s and 500s mean errors
else if ($info['http_code'] >= 400) {
    print "The server says HTTP error {$info['http_code']}. \n";
}
else {
    print "A successful result!\n";
}

// The request info includes timing statistics as well
print "By the way, this request took {$info['total_time']} seconds.\n";
```

# Using Cookies with cURL

By default, If the response to a cURL request includes a header that sets a cookie, **cURL ignores it**.

```
// Retrieve the cookie server page, sending no cookies
$c = curl_init('http://php7.example.com/cookie-server.php');
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);
// The first time, there are no cookies
$res = curl_exec($c);
print $res;

// The second time, there are still no cookies
$res = curl_exec($c);
print $res;
```

Enable **cURL's cookie jar** to keep track of cookies. set **CURLOPT\_COOKIEJAR** to **true**

- In this mode, the cookie jar only tracks cookies within a handle.

```
// Retrieve the cookie server page, sending no cookies
$c = curl_init('http://php7.example.com/cookie-server.php');
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);
// Turn on the cookie jar
curl_setopt($c, CURLOPT_COOKIEJAR, true);

// The first time, there are no cookies
$res = curl_exec($c);
print $res;

// The second time, there are cookies from the first request
$res = curl_exec($c);
print $res;
```

# Serving API Requests

---

- ⋮ Your PHP program can serve API requests to clients.
- ⋮ You can generate **data** for the client, also **HTTP response header**
- ⋮ To send an HTTP header along with your response, use the **header()**.

*Example 11-15. Serving a JSON response*

```
$response_data = array('now' => time());  
header('Content-Type: application/json');  
print json_encode($response_data);
```

```
HTTP/1.1 200 OK  
Host: www.example.com  
Connection: close  
Content-Type: application/json
```

```
{"now":1962258300}
```



# Exercises

---

1. <http://php.net/releases/?json> is a JSON feed of the latest PHP releases. Write a program that uses `file_get_contents()` to retrieve this feed and print out the latest version of PHP released.
2. Modify your program from the previous exercise to use cURL instead of `file_get_contents()`.
3. Write a web page that uses a cookie to tell the user when he last looked at the web page (you may find the date-and-time-handling functions described in [Chapter 15](#) useful).
4. A GitHub *gist* is a snippet of text or code that is easy to share. The [GitHub API](#) allows you to create gists without logging in. Write a program that creates a gist whose contents are the program you're writing to create a gist. Note that the GitHub API requires you to set a `User-Agent` header in your HTTP API requests. The `CURLOPT_USERAGENT` setting can be used to set this header.