

# Web Service: an Example

Alessio Vecchio

# Web Services: an Example

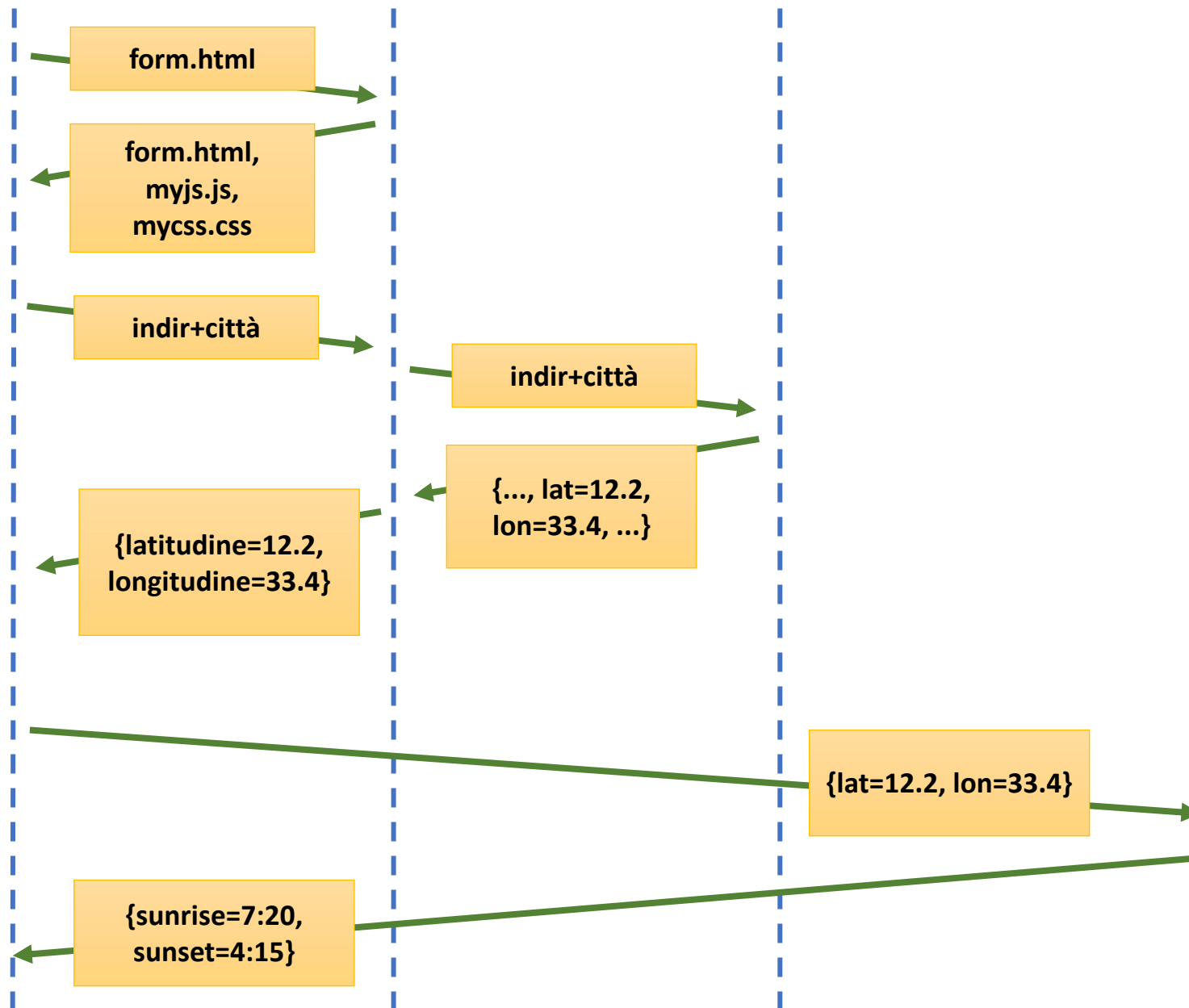
- An example about consuming web services
- User inserts address and city
- System returns sunrise and sunset times
- Two Web services used:
  - nominatim, provided by OSM, used for geocoding
  - sunrise-sunset.org, used for obtaining the times
- nominatim: used by PHP code (service.php)
- sunrise-sunset: used by JS code (myjs.js)

Browser

service.php

Nominatim

Sunrise-sunset




# form.html

```
<!DOCTYPE html>
<html lang="it">
<head>
  <title>Usare un web service</title>
  <link rel="stylesheet" href="mycss.css">
  <script src="myjs.js"></script>
</head>
<body onload="inizia()">
  <form>
    <div class="r">
      <label for="indirizzo" class="cell">Indirizzo</label>
      <input type="text" id="indirizzo" name="indirizzo"
        class="cell" pattern="[A-Za-z\s]{2,}"
        placeholder="via Mazzini" required>
    </div>
    <div class="r">
      <label for="citta" class="cell">Città</label>
      <input type="text" id="citta" name="citta"
        class="cell" pattern="^[A-Za-z\s]{2,}$"
        placeholder="Pisa" required>
    </div>
    <div class="r">
      <button id="invia" type="button" class="cell">Invia</button>
    </div>
  </form>
</body>
</html>
```

```
body {  
    background-color: azure;  
    font: 1em sans-serif;  
    padding: 1em;  
}  
  
form {  
    background-color: lavender;  
    width: fit-content;  
    height: fit-content;  
    border-spacing: 1em;  
}  
  
.r {  
    display: table-row;  
}  
  
.cell {  
    display: table-cell;  
}  
  
input:invalid {  
    background-color: orange;  
}
```

mycss.css



A screenshot of a web form with a lavender background. It contains two labels, 'Indirizzo' and 'Città', each followed by an orange input field. The first input field contains the text 'via Mazzini' and the second contains 'Pisa'. Below these fields is a button labeled 'Invia'.



A screenshot of a web form with a lavender background, similar to the one above but with white input fields. The labels 'Indirizzo' and 'Città' are followed by white input fields containing 'via Diotisalvi' and 'Pisa' respectively. A button labeled 'Invia' is positioned below the fields.

```
// URL del webservice che fornisce l'ora di alba e tramonto  
// prende in ingresso lat-lon
```

```
const url_sunrise = 'https://api.sunrise-sunset.org/json?';  
let bottone;
```

```
// Eseguita al caricamento della pagina
```

```
function inizia() {  
    bottone = document.getElementById('invia');  
    bottone.addEventListener('click', comincia);  
}
```

```
// Aggiunge al body un paragrafo contenente il messaggio passato come  
argomento
```

```
function aggiungiMessaggio(msg) {  
    const bod = document.getElementsByTagName('body')[0];  
    const par = document.createElement('p');  
    const txt = document.createTextNode(msg);  
    par.appendChild(txt);  
    bod.appendChild(par);  
}
```

```
// Eseguita quando l'utente preme il bottone Invia
```

```
function comincia() {  
    bottone.disabled = true;  
    inviaDati().then(console.log('fatto'));  
}
```

```
// Invia indirizzo e città al lato server
// che restituisce le coordinate lat-lon corrispondenti (geocoding)
async function inviaDati(){
  // Indirizzo e città dagli elementi input
  const indir = document.getElementById('indirizzo').value;
  const cit = document.getElementById('citta').value;
  // Query string
  const url = 'service.php?' + 'indirizzo='+indir+"&citta="+cit;
  try {
    let r = await fetch(url);          // Attende risposta
    let d = await r.json();           // Oggetto da JSON
    // Mostra coordinate all'utente
    aggiungiMessaggio("Coordinate: lat=" + d.latitudine +
                      ", lon=" + d.longitudine);
    // Query string per sunrise-sunset.org
    const searchurl = url_sunrise + "lat="+d.latitudine +
                      "&lon="+d.longitudine;
    let r2 = await fetch(searchurl); // Attende risposta
    let d2 = await r2.json();        // Oggetto da JSON
    // Mostra orari all'utente
    aggiungiMessaggio("Alba: " + d2.results.sunrise +
                      ", tramonto:" + d2.results.sunset);
  } catch (e) {
    aggiungiMessaggio("Sembra esserci un problema, riprova tra poco.");
  } finally {
    bottone.disabled = false;
  }
}
```

myjs.js

# service.php

```
<?php
// URL del servizio di geocoding di openstreetmap
// geocoding: indirizzo+citta -> lat-lon
define('NOMINATIM_URL',
        'https://nominatim.openstreetmap.org/search.php?');
// Controlla che indirizzo e citta siano presenti e non vuoti
if (isset($_GET['indirizzo']) && !empty($_GET['indirizzo'])
    &&
    isset($_GET['citta']) && !empty($_GET['citta'])) {
// Crea query string
$querystring =
    "city={$_GET['citta']}&street={$_GET['indirizzo']}" .
    "&format=json&email=alessio.vecchio@unipi.it";
// Sostituisce spazi con %20
$querystring = str_replace(' ', '%20', $querystring);
// Genera URL completo
$myurl = NOMINATIM_URL . $querystring;
```



```
// Inizializza curl
$mycurl = curl_init($myurl);
// Restituisce il risultato di curl_exec come una stringa
// invece di inviarlo direttamente in output
curl_setopt($mycurl, CURLOPT_RETURNTRANSFER, true);
// Non include gli header HTTP nell'output
curl_setopt($mycurl, CURLOPT_HEADER, false);
$risposta = curl_exec($mycurl); // Interroga il web service nominativim

if($risposta === false) {
    echo "Errore: ". curl_error($mycurl) .
        ", codice: " . curl_errno($mycurl);
} else {
    // Converte stringa JSON della risposta in un oggetto
    // L'oggetto è in effetti un array
    $jsonrisp = json_decode($risposta);
    // Crea risposta per il client come un array associativo
    $myres = array('latitudine' => $jsonrisp[0]->lat,
        'longitudine' => $jsonrisp[0]->lon);
    $x = json_encode($myres); // Codifica la risposta in JSON
    // Invia la risposta al client (browser)
    header("Content-type: application/json");
    echo $x;
}
// Libera risorse (non strettamente necessario da PHP 8.0)
curl_close($mycurl);
}
```

## Parameters

The search API has the following format:

```
https://nominatim.openstreetmap.org/search?<params>
```

The search term may be specified with two different sets of parameters:

- `q=<query>`

Free-form query string to search for. Free-form queries are processed first left-to-right and then right-to-left if that fails. So you may search for [pilkington avenue, birmingham](#) as well as for [birmingham, pilkington avenue](#). Commas are optional, but improve performance by reducing the complexity of the search.

- `street=<houzenumber> <streetname>`
- `city=<city>`
- `county=<county>`
- `state=<state>`
- `country=<country>`
- `postalcode=<postalcode>`

Alternative query string format split into several parameters for structured requests are faster but are less robust against alternative OSM tagging schemas. **combine with `q=<query>` parameter.**

Both query forms accept the additional parameters listed below.

## Output format

- `format=[xml|json|jsonv2|geojson|geocodejson]`

## JSON with address details

<https://nominatim.openstreetmap.org/?addressdetails=1&q=bakery+in+berlin+wedding&format=json&limit=1>

```
{
  "address": {
    "bakery": "B\u00e4cker Kamps",
    "city_district": "Mitte",
    "continent": "European Union",
    "country": "Deutschland",
    "country_code": "de",
    "footway": "Bahnsteig U6",
    "neighbourhood": "Sprengelkiez",
    "postcode": "13353",
    "state": "Berlin",
    "suburb": "Wedding"
  },
  "boundingbox": [
    "52.5460929870605",
    "52.5460968017578",
    "13.3591794967651",
    "13.3591804504395"
  ],
  "class": "shop",
  "display_name": "B\u00e4cker Kamps, Bahnsteig U6, Sprengelkiez, Wedding, Mitte, Berlin",
  "icon": "https://nominatim.openstreetmap.org/images/mapicons/shopping_bakery.p.20.png",
  "importance": 0.201,
  "lat": "52.5460941",
  "licence": "Data \u00a9 OpenStreetMap contributors, ODbL 1.0. https://www.openstreetmap.org/help/en#copyright",
  "lon": "13.35918",
  "osm_id": "317179427",
  "osm_type": "node",
  "place_id": "1453068",
  "type": "bakery"
}
```

# Sunset and sunrise times API

We offer a **free API** that provides **sunset and sunrise times** for a given **latitude and longitude**.

Please note that **attribution is required** if you use our API. Check "Usage limits and attribution" section below for more information.

## API documentation

Ours is a very simple REST api, you only have to do a GET request to **<https://api.sunset.org/json>**.

### Parameters

- **lat** (float): Latitude in decimal degrees. Required.
- **lng** (float): Longitude in decimal degrees. Required.
- **date** (string): Date in YYYY-MM-DD format. Also accepts [other date formats](#). If not present, date defaults to current date. Optional.
- **callback** (string): Callback function name for JSONP response. Optional.
- **formatted** (integer): 0 or 1 (1 is default). Time values in response will be expressed following [ISO 8601](#) and day\_length will be expressed in seconds. Optional.

### Sample requests

These are three sample requests for getting sunset and sunrise information from given location:

```
https://api.sunrise-sunset.org/json?lat=36.7201600&lng=-4.4203400&date=2023-01-01
https://api.sunrise-sunset.org/json?lat=36.7201600&lng=-4.4203400&date=2023-01-01&callback=jQuery
https://api.sunrise-sunset.org/json?lat=36.7201600&lng=-4.4203400&date=2023-01-01&callback=jQuery
```

### Response

The result data is formatted using JSON. An example:

**NOTE: All times are in UTC and summer time adjustments are not included in the returned data.**

```
{
  "results": {
    "sunrise": "7:27:02 AM",
    "sunset": "5:05:55 PM",
    "solar_noon": "12:16:28 PM",
    "day_length": "9:38:53",
    "civil_twilight_begin": "6:58:14 AM",
    "civil_twilight_end": "5:34:43 PM",
    "nautical_twilight_begin": "6:25:47 AM",
    "nautical_twilight_end": "6:07:10 PM",
    "astronomical_twilight_begin": "5:54:14 AM",
    "astronomical_twilight_end": "6:38:43 PM"
  },
  "status": "OK"
}
```

Indirizzo

Città

Coordinate: lat=43.7219374, lon=10.3881844

Alba: 7:16:17 AM, tramonto:4:24:53 PM