

Mapy google są coraz popularniejsze w internecie. Ich dokumentację można znaleźć pod adresem: <http://www.google.com/apis/maps/documentation/>

Niestety w dokumentacji tej brakuje bardzo przydatnej funkcji – otrzymywania nazwy miejscowości, gdy znane są współrzędne geograficzne. Na szczęście są możliwości rozwiązania tego problemu.

Opcja ta nazywa się geocode reverse i jest udostępniana przez różne firmy np. <http://ws.geonames.org/findNearbyPlaceName?lat=52&lng=20&maxRows=1>

### Wynikowy kod XML:

```
{codecitation class='brush: xml'}<geonames>    <geoname>
<toponymName>Jacochów</toponymName>    <name>Jacochów</name>
<lat>51.98333</lat>    <lng>19.98333</lng>    <geonameId>3097790</geonameId>
    <countryCode>PL</countryCode>    <countryName>Poland</countryName>
<fcl>P</fcl>    <fcode>PPL</fcode>    <distance>2.17633</distance>    </geoname>
</geonames>{/codecitation}
```

Zamiast kodu XML można uzyskać [kod JSON](http://ws.geonames.org/findNearbyPlaceNameJSON?lat=52&lng=20) - <http://ws.geonames.org/findNearbyPlaceNameJSON?lat=52&lng=20> . [patrz [klas a ObjTree](#)

]

Oczywiście parametry lat oraz lng a także wartość maxRows można zmieniać wg potrzeb. Najczęściej kiedy chcesz odczytać lokalizację wielu miejscowości po kolei. Warto wtedy trochę zautomatyzować działanie.

Ostatnio miałem przyjemność :P popracować z mapami google i stworzyłem taką o to klasę, pomagającą odczytać dane z serwera. Po odczytaniu przechowuje ona wartości pobrane z serwera w swoich właściwościach:

```
{codecitation class='brush: php'}
```

```
// class geocodeReverse // autor Patryk yarpo Jar // yarpo@poczta.onet.pl class
geocodeReverse { private $x; private $y; private $countryCode; private $contryName;
private $city; private $distance; private $maxRows; private $fileSize; private $xmlCode; //
x - float, // y - float, // $rows - unsigned int, max numbers of cities nearby this location
function __construct($x, $y, $rows) { $this->x = $x; $this->y = $y; $this->maxRows = $rows;
$this->fileSize = 1024; $this->changeLocation($x, $y); } public function getCity() { return
$this->city; } public function getXpos() { return $this->x; } public function getYpos() { return
$this->y; } public function getLng() { return $this->x; } public function getLat() { return
$this->y; } public function getCountryCode() { return $this->countryCode; } public function
getCountryName() { return $this->countryName; } public function getDistance() { return
$this->distance; } public function getMaxRows() { return $this->maxRows; } public function
getFileSize() { return $this->fileSize; } public function setMaxRows($newValue) {
$this->maxRows = $newValue; } public function setFileSize($newValue) { $this->fileSize =
$newValue; } public function changeLocation($x, $y) { $this->x = $x; $this->y = $y; $url =
'http://ws.geonames.org/findNearbyPlaceName?lat='.$y.'&lng='.$x.'&maxRows='.$this->maxRo
ws; $file = fopen($url, 'r'); $this->xmlCode = fread($file, $this->fileSize); fclose($file); //
create an object from XML file (string) $xml = new SimpleXMLElement($this->xmlCode);
$this->city = $xml->geoname->name; $this->x = $xml->geoname->lng; $this->y =
$xml->geoname->lat; $this->countryCode = $xml->geoname->countryCode;
$this->countryName = $xml->geoname->countryName; $this->distance =
$xml->geoname->distance; } public function getXMLCode() { return $this->xmlCode; } }; //
end of class
```

```
{/codecitation}
```

## Przykładowe zastosowanie

```
{codecitation class='brush: php'}
```

```
$geo = new geocodeRevers(52, 20, 1); echo $geo->getCountryCode();
```

```
{/codecitation}
```