PHP: effettuare richieste WHOIS

In questo tutorial vedremo come effettuare delle richieste WHOIS con PHP.

Implementeremo una soluzione basata sui socket per evitare il pericolo dell'inserimento di comandi arbitrari sulla shell. Creiamo per cominciare un file JSON contenente i dati dei server WHOIS con l'estensione supportata, il nome dell'host e la stringa restituita dal server nel caso in cui il dominio sia disponibile.

Ecco un esempio di un tale record.

```
{
    "extension": ".it",
    "server": "whois.nic.it",
    "response": "AVAILABLE"
},
```

Definiamo la seguente classe PHP.

```
class WHOIS {
    protected $domain;
    protected $servers;

    public function __construct($domain) {
        $this->domain = $domain;
        $this->servers = $this->getServerList();
    }

    public function lookup() {
        $output = [
            'error' => '',
            'response' => ''
        ];

        $extension = $this->getDomainExtension();

        if(!filter_var($this->domain, FILTER_VALIDATE_DOMAIN) || !$extension
|| strlen($extension) <= 2) {
            $output['error'] = 'Invalid domain name.';
            return $output;
        }

        $server = $this->getWhoisServer($extension);

        if(count($server) === 0) {
```

```
            $output['error'] = 'Invalid domain extension.';
            return $output;
        }

        $response = $this->request($server['server']);

        if(!$response) {
            $output['error'] = 'Connection error.';
            return $output;
        }

        $is_available = stristr($response, $server['response']) !== false;

        $output['response'] = $is_available ? 'Available' : 'Not Available';

        return $output;
    }

    private function getWhoisServer($extension) {
        $server = [];

        if(is_null($this->servers)) {
            return $server;
        }

        foreach($this->servers as $srv) {
            if($srv['extension'] === $extension) {
                $server = $srv;
            }
        }

        return $server;
    }

    private function getDomainExtension() {
        if(strpos($this->domain, '.') === false) {
            return false;
        }
        return substr($this->domain, strpos($this->domain, '.'));
    }

    private function getServerList() {
        $file = file_get_contents(dirname(__DIR__) . '/data/servers.json');
        if(!$file) {
            return null;
        }
        return json_decode($file, true);
    }

    private function request($server) {
        $response = "";
        $conn = fsockopen($server, 43, $errno, $errstr, 30);
        if(!$conn) {
```

```
            return false;
        }
        fputs($conn, $this->domain . "\r\n");
        while (!feof($conn)) {
            $response .= fread($conn,128);
        }
        fclose ($conn);
        return $response;
    }
}
```

Il metodo `lookup()` effettua una validazione preliminare del nome a dominio tenendo conto delle incoerenze di PHP nella validazione dei nomi a dominio. Quindi usa l'estensione del dominio per trovare il server WHOIS nella lista di quelli disponibili ed effettua la richiesta.

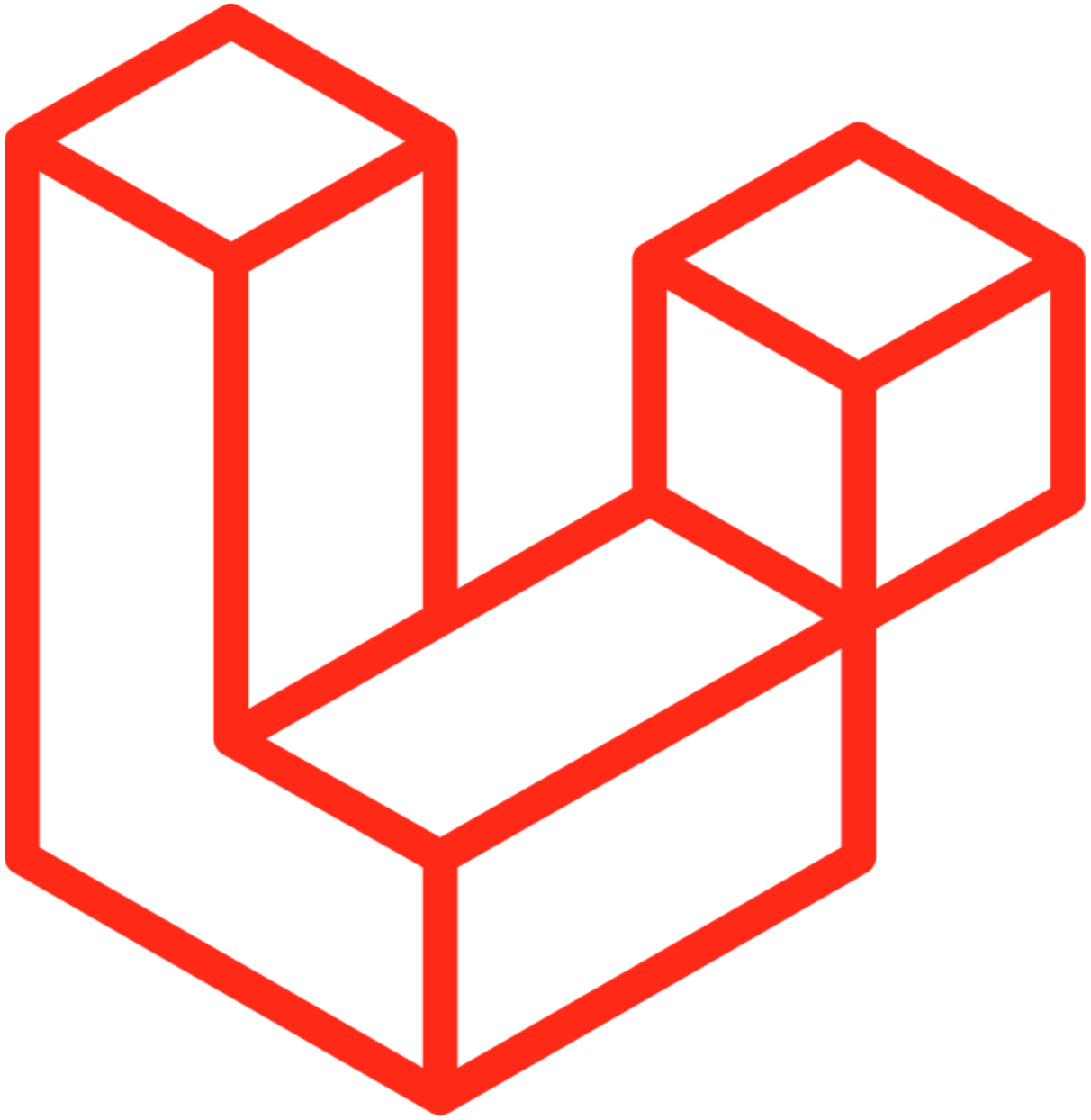
Possiamo usare la nostra classe in un endpoint AJAX nel modo seguente:

```
require_once 'src/WHOIS.php';
header('Content-Type: application/json');

$domain = trim($_POST['domain']);
$whois = new WHOIS($domain);
$output = $whois->lookup();
echo json_encode($output);
exit;
```

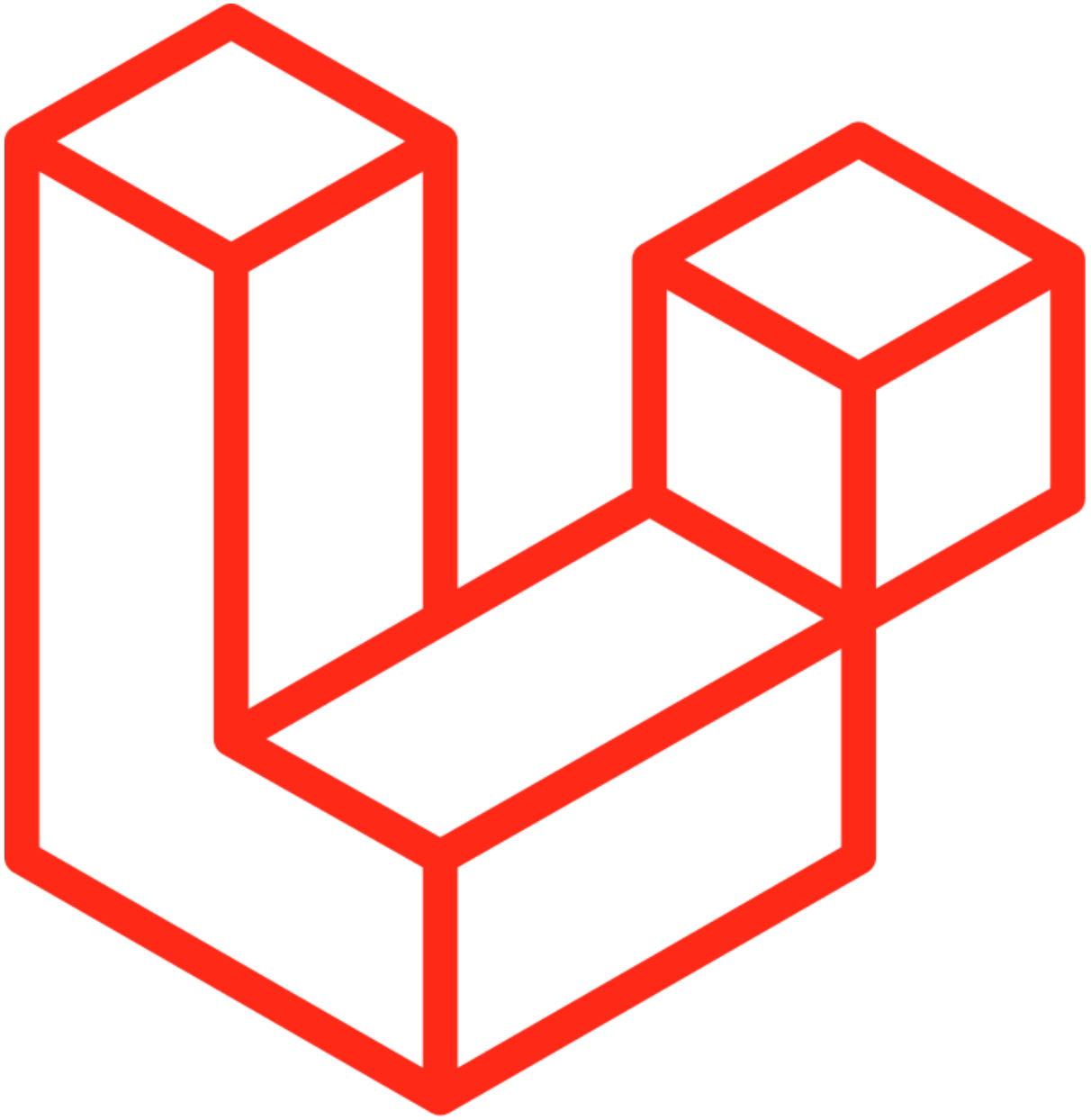
Si tratta di una soluzione che può essere adattata a molti contesti dello sviluppo con PHP.

Applicazioni Correlate



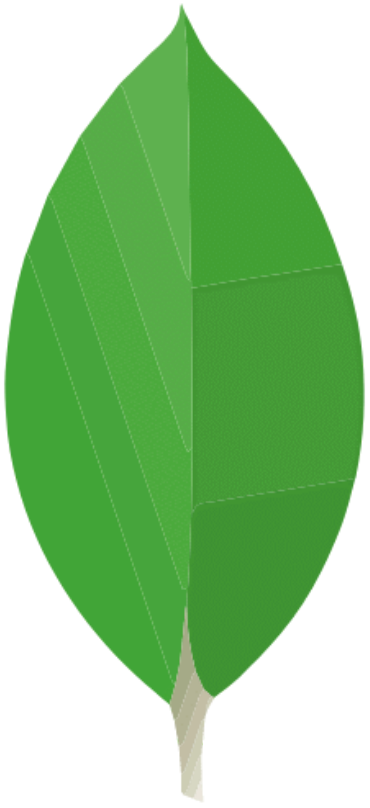
Laravel Placeholder Image

Applicazione in Laravel per creare immagini segnaposto.
DockerDocker ComposeComposerPHPLaravelJavaScript



Laravel Shopping Cart

Applicazione che fa parte del progetto Laravel E-commerce e implementa la gestione del carrello in Laravel.
DockerDocker ComposeComposerPHPLaravel



PHP MongoDB App

Applicazione basata su MongoDB con il driver PHP ufficiale.
DockerDocker ComposeComposerPHPMongoDB