Symfony has a wonderful `onKernelException` event that we can expose in an Event Listener to properly handle exceptions in our applications.

In one particular use case, we want to handle JSON and XML responses for our exceptions that are thrown via our API. In this case, we need to format and render our exception based on the Accept Header of the initial request.

This is handled in two simple steps:

1. An ApiExceptionListener class that processes the onKernelException event.
2. A service declaration that tells the Symfony kernel about our Event Listener.

The Exception Listener

```
<?php

namespace MyBundle\EventListener;

use Symfony\Component\HttpKernel\Event\GetResponseForExceptionEvent;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpKernel\Exception\HttpExceptionInterface;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\AcceptHeader;
use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;

class ApiExceptionListener
{

    /**
     * Respond with a properly formatted Exception when requesting via
the API.
     *
```

```
     * @param
\Symfony\Component\HttpKernel\Event\GetResponseForExceptionEvent
$event
     */
    public function onKernelException(GetResponseForExceptionEvent
$event)
    {
        // You get the exception object from the received event
        $exception = $event->getException();

        /**
         * @TODO: Handle Api Exception classes uniquely?
         */

        // HttpExceptionInterface is a special type of exception that
        // holds status code and header details
        if ($exception instanceof HttpExceptionInterface) {
            $statusCode = $exception->getStatusCode();
        }
        else {
            $statusCode = Response::HTTP_INTERNAL_SERVER_ERROR;
        }

        $request = $event->getRequest();
        $accept =
AcceptHeader::fromString($request->headers->get('Accept'));

        /**
         *  If the original request is accepting application/json or
if it was itself
         *  an XML request, then the response should be JSON
```

```
         */

        $data = array(
            'code'    => $statusCode,
            'message' => str_replace('"', "'",
$exception->getMessage()),
        );

        if( $accept->has('application/json') ) {
            $response = new Response(json_encode($data,
JSON_FORCE_OBJECT | JSON_UNESCAPED_SLASHES | JSON_UNESCAPED_UNICODE));
            $response->headers->set('Content-Type', 'application/json;
charset=utf-8');
            $event->setResponse($response);
        }
        elseif( $request->isXmlHttpRequest() ) {
            $xml = new \SimpleXMLElement('<exception/>');
            $data = array_flip($data);
            array_walk_recursive($data, array ($xml, 'addChild'));
            $response = new Response($xml->asXML());
            $response->headers->set('Content-Type', 'text/xml;
charset=utf-8');
            $event->setResponse($response);
        }
    }
}
```

We need to be careful above not to check the Accept header for XML because some browsers will give a false positive on this. As such, we check instead for an AJAX request that was not JSON, which thus must be XML.

Defining the service

```
api_exception_handler:
        class: MyBundle\EventListener\ApiExceptionListener
        tags:
            - { name: kernel.event_listener, event: kernel.exception,
method: onKernelException }
```

Sample Response

Now that we've got this setup, our exception messages are nice and clean:

```
{'code':500, 'message': 'We were unable to process your request!'}
```